RM0481
Reference manual

STM32H523/33xx, STM32H562/63xx, and STM32H573xx Arm®-based 32-bit MCUs

Introduction

This document is addressed to application developers. It provides complete information on how to use the STM32H523xx, STM32H533xx, STM32H562xx, STM32H563xx, and STM32H573xx microcontrollers memory and peripherals.

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

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

Related documents

- STM32H523/533xx datasheet
- STM32H562/563xx datasheet
- STM32H573xx datasheet
- STM32H523/533xx errata sheet
- STM32H562/563/573xx errata sheets
- STM32 Cortex®-M33 MCUs programming manual

Contents

1 Documentation conventions 104

1.1 General information 104

1.2 List of abbreviations for registers 104

1.3 Register reset value 104

1.4 Glossary 105

2 Memory and bus architecture 106

2.1 System architecture 106

2.1.1 Fast C-bus 107

2.1.2 Slow C-bus 108

2.1.3 S-bus 108

2.1.4 DCache S-bus 108

2.1.5 GPDMA1 and GPDMA2 buses 108

2.1.6 SDMMC1 and SDMMC2 controllers DMA buses 108

2.1.7 BusMatrix 108

2.1.8 AHB/APB bridges 109

2.1.9 Ethernet MAC 109

2.2 TrustZone security architecture 109

2.2.1 Default TrustZone security state 109

2.2.2 TrustZone peripheral classification 110

2.3 Memory organization 115

2.3.1 Introduction 115

2.3.2 Memory map and register boundary addresses 116

2.3.3 Embedded SRAMs 127

2.3.4 Flash memory overview 127

2.3.5 Boot modes 127

3 System security 129

3.1 Key security features 129

3.2 Secure install 130

3.3 Secure boot 130

3.3.1 Unique boot entry 131

3.3.2 Immutable root of trust in system flash memory 132

3.4 Secure update . . . 132
3.5 Resource isolation using hide protection levels . . . 132
3.6 Resource isolation using TrustZone . . . 132
3.6.1 TrustZone security architecture . . . 133
3.6.2 Armv8-M security extension of Cortex-M33 . . . 133
3.6.3 Memory and peripheral allocation using IDAU/SAU . . . 134
3.6.4 Memory and peripheral allocation using GTZC . . . 136
3.6.5 Managing security in TrustZone-aware peripherals . . . 140
3.6.6 Activating TrustZone security . . . 146
3.6.7 Deactivating TrustZone security . . . 147
3.7 Other resources isolation . . . 147
3.7.1 Temporal isolation using secure hide protection (HDP) . . . 148
3.7.2 Resource isolation using Cortex privileged mode . . . 148
3.8 Secure execution . . . 152
3.8.1 Memory protection unit (MPU) . . . 152
3.8.2 Embedded flash memory write protection . . . 153
3.8.3 Tamper detection and response . . . 153
3.9 Secure storage . . . 155
3.9.1 Hardware secret key management . . . 156
3.9.2 Unique ID . . . 157
3.10 Crypto engines . . . 157
3.10.1 Crypto engines features . . . 157
3.10.2 Secure AES co-processor (SAES) . . . 158
3.10.3 On-the-fly decryption engine (OTFDEC) . . . 159
3.11 Product life cycle . . . 159
3.11.1 Product configurations and security services . . . 160
3.11.2 Life cycle management . . . 161
3.11.3 Recommended product settings . . . 164
3.12 Software intellectual property protection and collaborative development 164
3.12.1 Software intellectual property protection . . . 166
3.12.2 Software intellectual property protection with OTFDEC . . . 166
3.12.3 Other software intellectual property protections . . . 168

4 Boot modes . . . 169
4.1 STM32H523/62/63xx boot modes . . . 170
4.2 STM32H533/73xx boot modes . . . 171

5 Global TrustZone® controller (GTZC) 172

5.1 GTZC introduction 172

5.2 GTZC main features 172

5.3 GTZC implementation 174

5.4 GTZC functional description 176

5.4.1 GTZC block diagram 176

5.4.2 Illegal access definition 177

5.4.3 TrustZone security controller (TZSC) 178

5.4.4 Memory protection controller - block based (MPCBB) 179

5.4.5 TrustZone illegal access controller (TZIC) 180

5.4.6 Power-on/reset state 180

5.5 GTZC interrupts 180

5.6 GTZC1 TZSC registers 181

5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR) 181

5.6.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1) 181

5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2) 184

5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3) 186

5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1) 188

5.6.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2) 191

5.6.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3) 193

5.6.8 GTZC1 TZSC memory x subregion A watermark configuration register (GTZC1_TZSC_MPCWMxACFGR) 195

5.6.9 GTZC1 TZSC memory x subregion A watermark register (GTZC1_TZSC_MPCWMxAR) 196

5.6.10 GTZC1 TZSC memory x subregion B watermark configuration register (GTZC1_TZSC_MPCWMxBCFGR) 196

5.6.11 GTZC1 TZSC memory x subregion B watermark register (GTZC1_TZSC_MPCWMxBR) 197

5.6.12 GTZC1 TZSC register map 199

5.7 GTZC1 TZIC registers 201

5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1) 201

5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2) 203

5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3) 205

5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4) 207
5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1) 209
5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2) 212
5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3) 214
5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4) 216
5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1) 218
5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2) 220
5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3) 222
5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4) 224
5.7.13 GTZC1 TZIC register map 227
5.8 GTZC1 MPCBBz registers (z = 1 to 3) 229
5.8.1 GTZC1 SRAMz MPCBB control register (GTZC1_MPCBBz_CR) (z = 1 to 3) 229
5.8.2 GTZC1 SRAMz MPCBB configuration lock register 1 (GTZC1_MPCBBz_CFGLOCK1) (z = 1 to 3) 229
5.8.3 GTZC1 SRAMz MPCBB security configuration for super-block x register (GTZC1_MPCBBz_SECCFGRx) (z = 1 to 3) 230
5.8.4 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 3) 230
5.8.5 GTZC1 MPCBBz register map (z = 1 to 3) 231

6 RAMs configuration controller (RAMCFG) 232
6.1 Introduction 232
6.2 RAMCFG main features 232
6.3 RAMCFG functional description 232
6.3.1 Internal SRAMs features 232
6.3.2 Error code correction (SRAM2, SRAM3, BKPSRAM) 233
6.3.3 Write protection (SRAM2) 235
6.3.4 Software erase 236
6.4 RAMCFG low-power modes 236
6.5 RAMCFG interrupts 236
6.6 RAMCFG registers 237
6.6.1 RAMCFG memory x control register (RAMCFG_MxCR) 237
6.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER) . . . 237
6.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR) 238
6.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR) 239

6.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR) . . . 239
6.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR) . . . 239
6.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1) . . . 240
6.6.8 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2) . . . 240
6.6.9 RAMCFG memory 2 write protection register 3 (RAMCFG_M2WPR3) . . . 241
6.6.10 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR) . . . 241
6.6.11 RAMCFG memory x erase key register (RAMCFG_MxERKEYR) . . . 242
6.6.12 RAMCFG register map . . . 243

7 Embedded flash memory (FLASH) . . . 246

7.1 Introduction . . . 246
7.2 FLASH main features . . . 246
7.3 FLASH functional description . . . 247
7.3.1 FLASH block diagram . . . 247
7.3.2 FLASH signals . . . 248
7.3.3 Flash memory architecture and usage . . . 249
7.3.4 FLASH read operations . . . 251
7.3.5 FLASH program operations . . . 253
7.3.6 FLASH erase operations . . . 256
7.3.7 FLASH parallel operations . . . 259
7.3.8 FLASH error protections . . . 260
7.3.9 OTP and RO memory access . . . 260
7.3.10 Flash high-cycle data . . . 262
7.3.11 Flash bank swapping . . . 264
7.3.12 FLASH reset and clocks . . . 267
7.4 FLASH option bytes . . . 269
7.4.1 About option bytes . . . 269
7.4.2 Option bytes loading . . . 269
7.4.3 Option bytes modification . . . 269
7.4.4 Description of user and system option bytes . . . 272
7.4.5 Description of data protection option bytes . . . 273
7.4.6 Description of boot address option bytes . . . 274
7.4.7 Specific rules for modifying option bytes . . . 274
7.5 Option bytes key (OBK) management . . . 276

7.5.1 OBK loading . . . 276
7.5.2 OBK access per HDPL level . . . 277
7.5.3 OBK programming sequence . . . 277
7.5.4 OBK programming finite state machine . . . 279
7.5.5 OBK swap sector . . . 280
7.5.6 OBK alternate sector erase . . . 282
7.6 FLASH security and protections . . . 282
7.6.1 TrustZone security protection . . . 283
7.6.2 Hide protection (HDP) . . . 285
7.6.3 Block-based secure flash memory area protection . . . 287
7.6.4 Block-based privileged flash memory area protection . . . 288
7.6.5 Flash memory register privileged and unprivileged modes . . . 289
7.6.6 Flash memory banks attributes in case of bank swap . . . 290
7.6.7 Flash memory configuration protection . . . 291
7.6.8 Write protection . . . 292
7.6.9 Flash high-cycle data protections . . . 292
7.6.10 Life cycle management . . . 294
7.6.11 Product state transitions . . . 295
7.6.12 OBK protection . . . 297
7.6.13 One-time-programmable and read-only memory protections . . . 298
7.7 System memory . . . 299
7.7.1 Introduction . . . 299
7.7.2 RSS user functions . . . 299
7.8 FLASH low-power modes . . . 306
7.9 FLASH error management . . . 307
7.9.1 Introduction . . . 307
7.9.2 Non-secure write protection error (WRPERR) . . . 307
7.9.3 Secure write protection error (WRPERR) . . . 308
7.9.4 Non secure programming sequence error (PGSERR) . . . 309
7.9.5 Secure programming sequence error (PGSERR) . . . 310
7.9.6 Non-secure strobe error (STRBERR) . . . 311
7.9.7 Secure strobe error (STRBERR) . . . 311
7.9.8 Non-secure inconsistency error (INCERR) . . . 311
7.9.9 Secure inconsistency error (INCERR) . . . 312
7.9.10 Error correction code error (ECCC, ECCD) . . . 313
7.9.11 Illegal access (ILAFM/ILAP) . . . 314
7.9.12 Option byte change error (OPTCHANGEERR) . . . 314

7.9.13 Miscellaneous HardFault errors . . . 315
7.9.14 OBK error cases (OBKERR, OBKWERR) . . . 315
7.10 FLASH interrupts . . . 316
7.11 FLASH registers . . . 318
7.11.1 FLASH access control register (FLASH_ACR) . . . 318
7.11.2 FLASH non-secure key register (FLASH_NSKEYR) . . . 319
7.11.3 FLASH secure key register (FLASH_SECKEYR) . . . 319
7.11.4 FLASH option key register (FLASH_OPTKEYR) . . . 320
7.11.5 FLASH non-secure OBK key register (FLASH_NSOBKKEYR) . . . 320
7.11.6 FLASH secure OBK key register (FLASH_SECOBKKEYR) . . . 320
7.11.7 FLASH operation status register (FLASH_OPSR) . . . 321
7.11.8 FLASH option control register (FLASH_OPTCR) . . . 322
7.11.9 FLASH non-secure status register (FLASH_NSSR) . . . 323
7.11.10 FLASH secure status register (FLASH_SECSR) . . . 325
7.11.11 FLASH non-secure control register (FLASH_NSCR) . . . 327
7.11.12 FLASH secure control register (FLASH_SECCR) . . . 329
7.11.13 FLASH non-secure clear control register (FLASH_NSCCR) . . . 332
7.11.14 FLASH secure clear control register (FLASH_SECCCR) . . . 333
7.11.15 FLASH privilege configuration register (FLASH_PRIVCFGR) . . . 334
7.11.16 FLASH non-secure OBK configuration register (FLASH_NSOBKCFGR) . . . 334
7.11.17 FLASH secure OBK configuration register (FLASH_SECOBKCFGR) 335
7.11.18 FLASH HDP extension register (FLASH_HDPEXTR) . . . 336
7.11.19 FLASH option status register (FLASH_OPTSR_CUR) . . . 337
7.11.20 FLASH option status register (FLASH_OPTSR_PRG) . . . 339
7.11.21 FLASH non-secure EPOCH register (FLASH_NSEPOCHR_CUR) . . 340
7.11.22 FLASH secure EPOCH register (FLASH_SECEPOCHR_CUR) . . . 341
7.11.23 FLASH option status register 2 (FLASH_OPTSR2_CUR) . . . 341
7.11.24 FLASH option status register 2 (FLASH_OPTSR2_PRG) . . . 342
7.11.25 FLASH non-secure boot register (FLASH_NSBOOTR_CUR) . . . 343
7.11.26 FLASH non-secure boot register (FLASH_NSBOOTR_PRG) . . . 344
7.11.27 FLASH secure boot register (FLASH_SECBOOTR_CUR) . . . 344
7.11.28 FLASH secure boot register (FLASH_BOOTR_PRG) . . . 345
7.11.29 FLASH non-secure OTP block lock (FLASH_OTPBLR_CUR) . . . 346
7.11.30 FLASH non-secure OTP block lock (FLASH_OTPBLR_PRG) . . . 346
7.11.31 FLASH secure block based register for Bank1 (FLASH_SECBB1Rx) 347

7.11.32 FLASH privilege block based register for Bank1 (FLASH_PRIVBB1Rx) . . . 347
7.11.33 FLASH security watermark for Bank1 (FLASH_SECWM1R_CUR) . . 348
7.11.34 FLASH security watermark for Bank1 (FLASH_SECWM1R_PRG) . . 348
7.11.35 FLASH write sector group protection for Bank1 (FLASH_WRP1R_CUR) . . . 349
7.11.36 FLASH write sector group protection for Bank1 (FLASH_WRP1R_PRG) . . . 349
7.11.37 FLASH data sector configuration Bank1 (FLASH_EDATA1R_CUR) . . 350
7.11.38 FLASH data sector configuration Bank1 (FLASH_EDATA1R_PRG) . . 351
7.11.39 FLASH HDP Bank1 configuration (FLASH_HDP1R_CUR) . . . 351
7.11.40 FLASH HDP Bank1 configuration (FLASH_HDP1R_PRG) . . . 352
7.11.41 FLASH ECC correction register (FLASH_ECCCORR) . . . 352
7.11.42 FLASH ECC detection register (FLASH_ECCDETR) . . . 353
7.11.43 FLASH ECC data (FLASH_ECCDR) . . . 354
7.11.44 FLASH secure block-based register for Bank2 (FLASH_SECBB2Rx) 355
7.11.45 FLASH privilege block-based register for Bank2 (FLASH_PRIVBB2Rx) . . . 355
7.11.46 FLASH security watermark for Bank2 (FLASH_SECWM2R_CUR) . . 356
7.11.47 FLASH security watermark for Bank2 (FLASH_SECWM2R_PRG) . . 356
7.11.48 FLASH write sector group protection for Bank2 (FLASH_WRP2R_CUR) . . . 357
7.11.49 FLASH write sector group protection for Bank2 (FLASH_WRP2R_PRG) . . . 358
7.11.50 FLASH data sectors configuration Bank2 (FLASH_EDATA2R_CUR) . 358
7.11.51 FLASH data sector configuration Bank2 (FLASH_EDATA2R_PRG) . . 359
7.11.52 FLASH HDP Bank2 configuration (FLASH_HDP2R_CUR) . . . 359
7.11.53 FLASH HDP Bank2 configuration (FLASH_HDP2R_PRG) . . . 360
7.12 FLASH register map and reset values . . . 361

8 Instruction cache (ICACHE) . . . 366

8.1 ICACHE introduction . . . 366
8.2 ICACHE main features . . . 366
8.3 ICACHE implementation . . . 367
8.4 ICACHE functional description . . . 367
8.4.1 ICACHE block diagram . . . 368
8.4.2 ICACHE reset and clocks . . . 368
8.4.3 ICACHE TAG memory . . . 369

8.4.4 Direct-mapped ICACHE (1-way cache) . . . 370
8.4.5 ICACHE enable . . . 371
8.4.6 Cacheable and noncacheable traffic . . . 371
8.4.7 Address remapping . . . 372
8.4.8 Cacheable accesses . . . 374
8.4.9 Dual-master cache . . . 375
8.4.10 ICACHE security . . . 375
8.4.11 ICACHE maintenance . . . 375
8.4.12 ICACHE performance monitoring . . . 376
8.4.13 ICACHE boot . . . 376
8.5 ICACHE low-power modes . . . 376
8.6 ICACHE error management and interrupts . . . 377
8.7 ICACHE registers . . . 377
8.7.1 ICACHE control register (ICACHE_CR) . . . 377
8.7.2 ICACHE status register (ICACHE_SR) . . . 378
8.7.3 ICACHE interrupt enable register (ICACHE_IER) . . . 379
8.7.4 ICACHE flag clear register (ICACHE_FCR) . . . 379
8.7.5 ICACHE hit monitor register (ICACHE_HMONR) . . . 380
8.7.6 ICACHE miss monitor register (ICACHE_MMONR) . . . 380
8.7.7 ICACHE region x configuration register (ICACHE_CRRx) . . . 380
8.7.8 ICACHE register map . . . 382

9 Data cache (DCACHE) . . . 383
9.1 DCACHE introduction . . . 383
9.2 DCACHE main features . . . 383
9.3 DCACHE implementation . . . 384
9.4 DCACHE functional description . . . 384
9.4.1 DCACHE block diagram . . . 385
9.4.2 DCACHE reset and clocks . . . 385
9.4.3 DCACHE TAG memory . . . 386
9.4.4 DCACHE enable . . . 388
9.4.5 Cacheable and noncacheable traffic . . . 388
9.4.6 Cacheable accesses . . . 389
9.4.7 DCACHE security . . . 391
9.4.8 DCACHE maintenance . . . 391
9.4.9 DCACHE performance monitoring . . . 393

9.4.10 DCACHE boot . . . 393
9.5 DCACHE low-power modes . . . 394
9.6 DCACHE error management and interrupts . . . 394
9.7 DCACHE registers . . . 395
9.7.1 DCACHE control register (DCACHE_CR) . . . 395
9.7.2 DCACHE status register (DCACHE_SR) . . . 396
9.7.3 DCACHE interrupt enable register (DCACHE_IER) . . . 397
9.7.4 DCACHE flag clear register (DCACHE_FCR) . . . 398
9.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR) . . . 398
9.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR) . . . 399
9.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR) . . . 399
9.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR) . . . 399
9.7.9 DCACHE command range start address register (DCACHE_CMDRSADDRR) . . . 400
9.7.10 DCACHE command range end address register (DCACHE_CMDREADDRR) . . . 400
9.7.11 DCACHE register map . . . 400

10 Power control (PWR) . . . 402
10.1 Introduction . . . 402
10.2 PWR main features . . . 402
10.3 PWR pins and internal signals . . . 403
10.4 PWR power supplies and supply domains . . . 404
10.4.1 External power supplies . . . 405
10.4.2 Internal regulators . . . 406
10.4.3 Power-up and power-down power sequences . . . 408
10.4.4 Independent analog peripherals supply . . . 408
10.4.5 Independent I/O supply rail . . . 409
10.4.6 Independent USB transceivers supply . . . 409
10.4.7 Backup domain . . . 409
10.5 PWR system supply voltage regulation . . . 411
10.5.1 SMPS and LDO embedded regulators . . . 411
10.5.2 V_{CORE} supply versus reset, voltage scaling, and low-power modes . . . 411
10.5.3 Embedded voltage regulator operating modes . . . 411
10.6 PWR power supply and temperature supervision . . . 412
10.6.1 Power-on reset (POR)/power-down reset (PDR) . . . 412
10.6.2 Brownout reset (BOR) . . . 413

10.6.3 Programmable voltage detector (PVD) . . . 414
10.6.4 Analog voltage detector (AVD) . . . 414
10.6.5 VDDIO2 voltage monitor (IO2VM) . . . 415
10.6.6 Backup domain voltage monitoring . . . 415
10.6.7 Temperature monitoring . . . 416
10.7 PWR management . . . 417
10.7.1 Voltage scaling . . . 417
10.7.2 Power management examples . . . 418
10.8 Power modes . . . 419
10.8.1 Slowing down system clocks . . . 423
10.8.2 Peripheral clock gating . . . 423
10.8.3 Low-power modes . . . 423
10.8.4 Sleep mode . . . 424
10.8.5 Stop mode . . . 425
10.8.6 Standby mode . . . 427
10.8.7 Power modes output pins . . . 430
10.9 PWR security and privileged protection . . . 431
10.9.1 PWR security protection . . . 431
10.9.2 PWR privileged protection . . . 432
10.10 PWR interrupts . . . 433
10.11 PWR registers . . . 434
10.11.1 PWR power mode control register (PWR_PMCR) . . . 434
10.11.2 PWR power mode control register [alternate] (PWR_PMCR) . . . 436
10.11.3 PWR status register (PWR_PMSR) . . . 437
10.11.4 PWR voltage scaling control register (PWR_VOSCR) . . . 438
10.11.5 PWR voltage scaling status register (PWR_VOSSR) . . . 439
10.11.6 PWR Backup domain control register (PWR_BDCR) . . . 439
10.11.7 PWR Backup domain control register (PWR_DBPCR) . . . 441
10.11.8 PWR Backup domain status register (PWR_BDSR) . . . 441
10.11.9 PWR USB Type-C power delivery register (PWR_UCPDR) . . . 442
10.11.10 PWR supply configuration control register (PWR_SCCR) . . . 442
10.11.11 PWR voltage monitor control register (PWR_VMCR) . . . 443
10.11.12 PWR USB supply control register (PWR_USBSCR) . . . 444
10.11.13 PWR voltage monitor status register (PWR_VMSR) . . . 445
10.11.14 PWR wake-up status clear register (PWR_WUSCR) . . . 446
10.11.15 PWR wake-up status register (PWR_WUSR) . . . 446

10.11.16 PWR wake-up configuration register (PWR_WUCR) 447
10.11.17 PWR I/O retention register (PWR_IORETR) 447
10.11.18 PWR security configuration register (PWR_SECCFGR) 448
10.11.19 PWR privilege configuration register (PWR_PRIVCFGR) 449
10.11.20 PWR register map 450

11 Reset and clock control (RCC) 452

11.1 Introduction 452
11.2 RCC pins and internal signals 452
11.3 RCC reset functional description 452
11.3.1 Power reset 452
11.3.2 System reset 453
11.3.3 Backup domain reset 454
11.3.4 Reset source identification 454
11.4 RCC clocks functional description 455
11.4.1 HSE clock 457
11.4.2 HSI clock 458
11.4.3 CSI oscillator 459
11.4.4 HSI48 clock 460
11.4.5 PLL description 460
11.4.6 LSE clock 464
11.4.7 LSI clock 465
11.4.8 System clock (SYSCLK) selection 465
11.4.9 Handling clock generators in stop and standby modes 466
11.4.10 Clock security system (CSS) 467
11.4.11 Clock output generation (MCO1/MCO2) 468
11.4.12 Kernel clock selection 468
11.4.13 RTC and TAMP clock 472
11.4.14 Timer clock 473
11.4.15 Watchdog clock 473
11.4.16 Peripherals clock gating and autonomous mode 473
11.5 RCC security and privilege functional description 474
11.5.1 RCC TrustZone security protection modes 474
11.5.2 RCC privilege protection modes 477
11.6 RCC low-power modes 477
11.7 RCC interrupts 479

11.8 RCC registers 481
11.8.1 RCC clock control register (RCC_CR) 481
11.8.2 RCC HSI calibration register (RCC_HSICFGR) 484
11.8.3 RCC clock recovery RC register (RCC_CRRCR) 484
11.8.4 RCC CSI calibration register (RCC_CSICFGR) 485
11.8.5 RCC clock configuration register1 (RCC_CFGR1) 485
11.8.6 RCC CPU domain clock configuration register 2 (RCC_CFGR2) 488
11.8.7 RCC PLL clock source selection register (RCC_PLL1CFGR) 490
11.8.8 RCC PLL clock source selection register (RCC_PLL2CFGR) 492
11.8.9 RCC PLL clock source selection register (RCC_PLL3CFGR) 494
11.8.10 RCC PLL1 dividers register (RCC_PLL1DIVR) 495
11.8.11 RCC PLL1 fractional divider register (RCC_PLL1FRACR) 497
11.8.12 RCC PLL1 dividers register (RCC_PLL2DIVR) 497
11.8.13 RCC PLL2 fractional divider register (RCC_PLL2FRACR) 498
11.8.14 RCC PLL3 dividers register (RCC_PLL3DIVR) 499
11.8.15 RCC PLL3 fractional divider register (RCC_PLL3FRACR) 500
11.8.16 RCC clock source interrupt enable register (RCC_CIER) 501
11.8.17 RCC clock source interrupt flag register (RCC_CIFR) 502
11.8.18 RCC clock source interrupt clear register (RCC_CICR) 504
11.8.19 RCC AHB1 reset register (RCC_AHB1RSTR) 505
11.8.20 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) 506
11.8.21 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) 508
11.8.22 RCC APB1 peripheral low reset register (RCC_APB1LRSTR) 509
11.8.23 RCC APB1 peripheral high reset register (RCC_APB1HRSTR) 512
11.8.24 RCC APB2 peripheral reset register (RCC_APB2RSTR) 513
11.8.25 RCC APB3 peripheral reset register (RCC_APB3RSTR) 515
11.8.26 RCC AHB1 peripherals clock register (RCC_AHB1ENR) 516
11.8.27 RCC AHB2 peripheral clock register (RCC_AHB2ENR) 518
11.8.28 RCC AHB4 peripheral clock register (RCC_AHB4ENR) 520
11.8.29 RCC APB1 peripheral clock register (RCC_APB1LENR) 521
11.8.30 RCC APB1 peripheral clock register (RCC_APB1HENR) 524
11.8.31 RCC APB2 peripheral clock register (RCC_APB2ENR) 525
11.8.32 RCC APB3 peripheral clock register (RCC_APB3ENR) 526
11.8.33 RCC AHB1 sleep clock register (RCC_AHB1LPENR) 528
11.8.34 RCC AHB2 sleep clock register (RCC_AHB2LPENR) 530
11.8.35 RCC AHB4 sleep clock register (RCC_AHB4LPENR) 532
11.8.36 RCC APB1 sleep clock register (RCC_APB1LLPENR) 533

11.8.37 RCC APB1 sleep clock register (RCC_APB1HLPENR) 536
11.8.38 RCC APB2 sleep clock register (RCC_APB2LPENR) 537
11.8.39 RCC APB3 sleep clock register (RCC_APB3LPENR) 539
11.8.40 RCC kernel clock configuration register (RCC_CCIPR1) 541
11.8.41 RCC kernel clock configuration register (RCC_CCIPR2) 543
11.8.42 RCC kernel clock configuration register (RCC_CCIPR3) 546
11.8.43 RCC kernel clock configuration register (RCC_CCIPR4) 547
11.8.44 RCC kernel clock configuration register (RCC_CCIPR5) 549
11.8.45 RCC Backup domain control register (RCC_BDCR) 551
11.8.46 RCC reset status register (RCC_RSR) 553
11.8.47 RCC secure configuration register (RCC_SECCFGR) 554
11.8.48 RCC privilege configuration register (RCC_PRIVCFGR) 556
11.9 RCC register map 557

12 Clock recovery system (CRS) 563
12.1 CRS introduction 563
12.2 CRS main features 563
12.3 CRS implementation 563
12.4 CRS functional description 564
12.4.1 CRS block diagram 564
12.4.2 CRS internal signals 564
12.4.3 Synchronization input 565
12.4.4 Frequency error measurement 565
12.4.5 Frequency error evaluation and automatic trimming 566
12.4.6 CRS initialization and configuration 567
12.5 CRS in low-power modes 568
12.6 CRS interrupts 568
12.7 CRS registers 568
12.7.1 CRS control register (CRS_CR) 568
12.7.2 CRS configuration register (CRS_CFGR) 570
12.7.3 CRS interrupt and status register (CRS_ISR) 571
12.7.4 CRS interrupt flag clear register (CRS_ICR) 573
12.7.5 CRS register map 573

13 General-purpose I/Os (GPIO) 575
13.1 Introduction 575

13.2 GPIO main features 575
13.3 GPIO functional description 575
13.3.1 General-purpose I/O (GPIO) 577
13.3.2 I/O pin alternate function multiplexer and mapping 578
13.3.3 I/O port control registers 578
13.3.4 I/O port data registers 579
13.3.5 I/O data bitwise handling 579
13.3.6 GPIO locking mechanism 579
13.3.7 I/O alternate function input/output 579
13.3.8 External interrupt/wake-up lines 580
13.3.9 Input configuration 580
13.3.10 Output configuration 581
13.3.11 Alternate function configuration 581
13.3.12 Analog configuration 582
13.3.13 Using the HSE or LSE oscillator pins as GPIOs 582
13.3.14 Using the GPIO pins in the RTC supply domain 583
13.3.15 I/Os state retention during standby mode 583
13.3.16 TrustZone security 583
13.3.17 Privileged and unprivileged modes 584
13.3.18 High-speed low-voltage mode (HSLV) 584
13.3.19 I/O compensation cell 585
13.4 GPIO registers 586
13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to I) 586
13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to I) 586
13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to I) 587
13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I) . . . 587
13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to I) 588
13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to I) 588
13.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I) 589
13.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to I) 589
13.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to I) 590
13.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H) . . . 591
13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to I) 592
13.4.12 GPIO high-speed low-voltage register (GPIOx_HSLVR) (x = A to I) . . 592
13.4.13 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to I) . 593
13.4.14 GPIO register map 594

14 System configuration, boot, and security (SBS) 596

14.1 SBS introduction 596

14.2 SBS main features 596

14.3 SBS functional description 597

14.3.1 SBS block diagram 597

14.3.2 SBS signals 598

14.3.3 SBS reset and clocks 598

14.3.4 SBS system configuration 599

14.3.5 SBS boot control 600

14.3.6 SBS debug control 603

14.3.7 SBS hardware secure storage control 605

14.4 SBS interrupts 607

14.5 SBS registers 607

14.5.1 SBS temporal isolation control register (SBS_HDPLCR) 607

14.5.2 SBS temporal isolation status register (SBS_HDPLSR) 607

14.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR) 608

14.5.4 SBS debug control register (SBS_DBGCR) 608

14.5.5 SBS debug lock register (SBS_DBGLOCKR) 609

14.5.6 SBS RSS command register (SBS_RSSCMDR) 610

14.5.7 SBS EPOCH selection control register (SBS_EPOCHSELCR) 610

14.5.8 SBS security mode configuration control register (SBS_SECCFGR) . 611

14.5.9 SBS product mode and configuration register (SBS_PMCR) 611

14.5.10 SBS FPU interrupt mask register (SBS_FPUIMR) 613

14.5.11 SBS memory erase status register (SBS_MESR) 613

14.5.12 SBS compensation cell for I/Os control and status register (SBS_CCCSR) 614

14.5.13 SBS compensation cell for I/Os value register (SBS_CCVALR) 615

14.5.14 SBS compensation cell for I/Os software code register (SBS_CCSWCR) 616

14.5.15 SBS Class B register (SBS_CFGR2) 616

14.5.16 SBS CPU nonsecure lock register (SBS_CNSLCKR) 617

14.5.17 SBS CPU secure lock register (SBS_CSLCKR) 618

14.5.18 SBS flift ECC NMI mask register (SBS_ECCNMIR) 619

14.5.19 SBS register map 620

15 Peripherals interconnect matrix 622

15.1 Interconnect matrix introduction 622

15.2 Connection summary 622
15.3 Interconnection details 623
15.3.1 Master to slave interconnection for timers 623
15.3.2 Triggers to ADCs 624
15.3.3 ADC analog watchdogs as triggers to timers 625
15.3.4 Triggers to DAC 625
15.3.5 Clock sources to timers 626
15.3.6 Triggers to low-power timers 627
15.3.7 RTC wake-up as inputs to timers 627
15.3.8 System errors as break signals to timers 627
15.3.9 Triggers for communication peripherals 628
15.3.10 Triggers to GPDMA1/2 628
15.3.11 Internal analog signals to analog peripherals 629
15.3.12 Clock source for the DAC sample and hold mode 629
15.3.13 Internal tamper sources 630
15.3.14 Output from tamper to RTC 630
15.3.15 Encryption keys to AES/SAES 630

16 General purpose direct memory access controller (GPDMA) 631
16.1 GPDMA introduction 631
16.2 GPDMA main features 631
16.3 GPDMA implementation 632
16.3.1 GPDMA instances 632
16.3.2 GPDMA channels 632
16.3.3 GPDMA in low-power modes 633
16.3.4 GPDMA requests 633
16.3.5 GPDMA block requests 638
16.3.6 GPDMA channels with peripheral early termination 638
16.3.7 GPDMA triggers 638
16.4 GPDMA functional description 641
16.4.1 GPDMA block diagram 641
16.4.2 GPDMA channel state and direct programming without any linked-list 641
16.4.3 GPDMA channel suspend and resume 642
16.4.4 GPDMA channel abort and restart 643
16.4.5 GPDMA linked-list data structure 644
16.4.6 Linked-list item transfer execution 647

16.4.7 GPDMA channel state and linked-list programming in run-to-completion mode . . . 648
16.4.8 GPDMA channel state and linked-list programming in link step mode 651
16.4.9 GPDMA channel state and linked-list programming . . . 658
16.4.10 GPDMA FIFO-based transfers . . . 660
16.4.11 GPDMA transfer request and arbitration . . . 667
16.4.12 GPDMA triggered transfer . . . 671
16.4.13 GPDMA circular buffering with linked-list programming . . . 672
16.4.14 GPDMA transfer in peripheral flow-control mode . . . 674
16.4.15 GPDMA secure/nonsecure channel . . . 675
16.4.16 GPDMA privileged/unprivileged channel . . . 676
16.4.17 GPDMA error management . . . 676
16.5 GPDMA in debug mode . . . 678
16.6 GPDMA in low-power modes . . . 678
16.7 GPDMA interrupts . . . 679
16.8 GPDMA registers . . . 680
16.8.1 GPDMA secure configuration register (GPDMA_SECCFGR) . . . 680
16.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR) . . . 681
16.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR) . . . 681
16.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR) . . . 682
16.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR) . . 683
16.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) . . . 683
16.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR) . . . 684
16.8.8 GPDMA channel x status register (GPDMA_CxSR) . . . 685
16.8.9 GPDMA channel x control register (GPDMA_CxCR) . . . 686
16.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1) . . . 688
16.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2) . . . 692
16.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1) . . . 696
16.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) . . . 697
16.8.14 GPDMA channel x source address register (GPDMA_CxSAR) . . . 700
16.8.15 GPDMA channel x destination address register (GPDMA_CxDAR) . . 702
16.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3) . . . 703
16.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2) . . . 704
16.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR) . . . 705
16.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) . . . 707

16.8.20 GPDMA register map 708

17 Nested vectored interrupt controller (NVIC) 711

17.1 NVIC main features 711

17.2 SysTick calibration value register 711

17.3 Interrupt and exception vectors 712

18 Extended interrupts and event controller (EXTI) 722

18.1 EXTI main features 722

18.2 EXTI block diagram 722

18.2.1 EXTI connections between peripherals and CPU 724

18.2.2 EXTI interrupt/event mapping 724

18.3 EXTI functional description 726

18.3.1 EXTI configurable event input wake-up 726

18.3.2 EXTI direct event input wake-up 727

18.3.3 EXTI mux selection 728

18.4 EXTI functional behavior 728

18.5 EXTI event protection 729

18.5.1 EXTI security protection 729

18.5.2 EXTI privilege protection 730

18.6 EXTI registers 731

18.6.1 EXTI rising trigger selection register (EXTI_RTSR1) 731

18.6.2 EXTI falling trigger selection register (EXTI_FTSR1) 731

18.6.3 EXTI software interrupt event register (EXTI_SWIER1) 732

18.6.4 EXTI rising edge pending register (EXTI_RPR1) 733

18.6.5 EXTI falling edge pending register (EXTI_FPR1) 733

18.6.6 EXTI security configuration register (EXTI_SECCFGR1) 734

18.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1) 735

18.6.8 EXTI rising trigger selection register 2 (EXTI_RTSR2) 735

18.6.9 EXTI falling trigger selection register 2 (EXTI_FTSR2) 736

18.6.10 EXTI software interrupt event register 2 (EXTI_SWIER2) 737

18.6.11 EXTI rising edge pending register 2 (EXTI_RPR2) 739

18.6.12 EXTI falling edge pending register 2 (EXTI_FPR2) 740

18.6.13 EXTI security configuration register 2 (EXTI_SECCFGR2) 741

18.6.14 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2) 742

18.6.15 EXTI external interrupt selection register (EXTI_EXTICR1) 742

18.6.16 EXTI external interrupt selection register (EXTI_EXTICR2) 744
18.6.17 EXTI external interrupt selection register (EXTI_EXTICR3) 746
18.6.18 EXTI external interrupt selection register (EXTI_EXTICR4) 748
18.6.19 EXTI lock register (EXTI_LOCKR) 750
18.6.20 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1) 751
18.6.21 EXTI CPU wake-up with event mask register (EXTI_EMR1) 751
18.6.22 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2) 752
18.6.23 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2) 753
18.6.24 EXTI register map 754

19 Cyclic redundancy check calculation unit (CRC) 756

19.1 CRC introduction 756
19.2 CRC main features 756
19.3 CRC functional description 757
19.3.1 CRC block diagram 757
19.3.2 CRC internal signals 757
19.3.3 CRC operation 757
19.4 CRC registers 759
19.4.1 CRC data register (CRC_DR) 759
19.4.2 CRC independent data register (CRC_IDR) 759
19.4.3 CRC control register (CRC_CR) 760
19.4.4 CRC initial value (CRC_INIT) 761
19.4.5 CRC polynomial (CRC_POL) 761
19.4.6 CRC register map 762

20 CORDIC coprocessor (CORDIC) 763

20.1 CORDIC introduction 763
20.2 CORDIC main features 763
20.3 CORDIC functional description 763
20.3.1 General description 763
20.3.2 CORDIC functions 764
20.3.3 Fixed point representation 770
20.3.4 Scaling factor 771
20.3.5 Precision 771
20.3.6 Zero-overhead mode 774
20.3.7 Polling mode 775

20.3.8 Interrupt mode . . . 776
20.3.9 DMA mode . . . 776
20.4 CORDIC registers . . . 777
20.4.1 CORDIC control/status register (CORDIC_CSR) . . . 777
20.4.2 CORDIC argument register (CORDIC_WDATA) . . . 779
20.4.3 CORDIC result register (CORDIC_RDATA) . . . 780
20.4.4 CORDIC register map . . . 780

21 Filter math accelerator (FMAC) . . . 781
21.1 FMAC introduction . . . 781
21.2 FMAC main features . . . 781
21.3 FMAC functional description . . . 782
21.3.1 General description . . . 782
21.3.2 Local memory and buffers . . . 783
21.3.3 Input buffers . . . 783
21.3.4 Output buffer . . . 786
21.3.5 Initialization functions . . . 788
21.3.6 Filter functions . . . 789
21.3.7 Fixed point representation . . . 793
21.3.8 Implementing FIR filters with the FMAC . . . 793
21.3.9 Implementing IIR filters with the FMAC . . . 795
21.3.10 Examples of filter initialization . . . 797
21.3.11 Examples of filter operation . . . 798
21.3.12 Filter design tips . . . 800
21.4 FMAC registers . . . 801
21.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG) . . . 801
21.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG) . . . 801
21.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG) . . . 802
21.4.4 FMAC parameter register (FMAC_PARAM) . . . 803
21.4.5 FMAC control register (FMAC_CR) . . . 804
21.4.6 FMAC status register (FMAC_SR) . . . 805
21.4.7 FMAC write data register (FMAC_WDATA) . . . 806
21.4.8 FMAC read data register (FMAC_RDATA) . . . 807
21.4.9 FMAC register map . . . 807

22 Flexible memory controller (FMC) . . . 809
22.1 Introduction . . . 809

22.2 FMC main features . . . 809
22.3 FMC block diagram . . . 810
22.4 AHB interface . . . 811
22.4.1 Supported memories and transactions . . . 812
22.5 External device address mapping . . . 813
22.5.1 NOR/PSRAM address mapping . . . 814
22.5.2 NAND flash memory address mapping . . . 814
22.5.3 SDRAM address mapping . . . 815
22.6 NOR flash/PSRAM controller . . . 817
22.6.1 External memory interface signals . . . 819
22.6.2 Supported memories and transactions . . . 821
22.6.3 General timing rules . . . 822
22.6.4 NOR flash/PSRAM controller asynchronous transactions . . . 822
22.6.5 Synchronous transactions . . . 840
22.6.6 NOR/PSRAM controller registers . . . 847
22.7 NAND flash controller . . . 855
22.7.1 External memory interface signals . . . 855
22.7.2 NAND flash supported memories and transactions . . . 856
22.7.3 Timing diagrams for NAND flash memory . . . 857
22.7.4 NAND flash operations . . . 858
22.7.5 NAND flash prewait functionality . . . 858
22.7.6 Computation of the error correction code (ECC) in NAND flash memory . . . 859
22.7.7 NAND flash controller registers . . . 860
22.8 SDRAM controller . . . 866
22.8.1 SDRAM controller main features . . . 866
22.8.2 SDRAM External memory interface signals . . . 866
22.8.3 SDRAM controller functional description . . . 867
22.8.4 Low-power modes . . . 873
22.8.5 SDRAM controller registers . . . 876
22.8.6 FMC register map . . . 882

23 Octo-SPI interface (OCTOSPI) . . . 885
23.1 OCTOSPI introduction . . . 885
23.2 OCTOSPI main features . . . 885
23.3 OCTOSPI implementation . . . 886

23.4 OCTOSPI functional description 887
23.4.1 OCTOSPI block diagram 887
23.4.2 OCTOSPI pins and internal signals 888
23.4.3 OCTOSPI interface to memory modes 889
23.4.4 OCTOSPI regular-command protocol 889
23.4.5 OCTOSPI regular-command protocol signal interface 893
23.4.6 HyperBus protocol 896
23.4.7 Specific features 900
23.4.8 OCTOSPI operating mode introduction 901
23.4.9 OCTOSPI indirect mode 901
23.4.10 OCTOSPI automatic status-polling mode 903
23.4.11 OCTOSPI memory-mapped mode 904
23.4.12 OCTOSPI configuration introduction 905
23.4.13 OCTOSPI system configuration 905
23.4.14 OCTOSPI device configuration 905
23.4.15 OCTOSPI regular-command mode configuration 908
23.4.16 OCTOSPI HyperBus protocol configuration 910
23.4.17 OCTOSPI error management 911
23.4.18 OCTOSPI BUSY and ABORT 911
23.4.19 OCTOSPI reconfiguration or deactivation 912
23.4.20 NCS behavior 912
23.5 Address alignment and data number 914
23.6 OCTOSPI interrupts 915
23.7 OCTOSPI registers 915
23.7.1 OCTOSPI control register (OCTOSPI_CR) 915
23.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1) 918
23.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2) 920
23.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3) 921
23.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4) 921
23.7.6 OCTOSPI status register (OCTOSPI_SR) 922
23.7.7 OCTOSPI flag clear register (OCTOSPI_FCR) 923
23.7.8 OCTOSPI data length register (OCTOSPI_DLR) 923
23.7.9 OCTOSPI address register (OCTOSPI_AR) 924
23.7.10 OCTOSPI data register (OCTOSPI_DR) 924
23.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR) 925
23.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR) 926
23.7.13 OCTOSPI polling interval register (OCTOSPI_PIR) 926

23.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR) . . 926
23.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR) 928
23.7.16 OCTOSPI instruction register (OCTOSPI_IR) 929
23.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR) 929
23.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR) 930
23.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR) 930
23.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR) . . 932
23.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR) 933
23.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR) 933
23.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR) 934
23.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR) 936
23.7.25 OCTOSPI write instruction register (OCTOSPI_WIR) 936
23.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR) 937
23.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR) 937
23.7.28 OCTOSPI register map 938

24 Secure digital input/output MultiMediaCard interface (SDMMC) . . . 941

24.1 SDMMC main features 941
24.2 SDMMC implementation 941
24.3 SDMMC bus topology 942
24.4 SDMMC operation modes 944
24.5 SDMMC functional description 945
24.5.1 SDMMC block diagram 945
24.5.2 SDMMC pins and internal signals 945
24.5.3 General description 946
24.5.4 SDMMC adapter 948
24.5.5 SDMMC AHB slave interface 970
24.5.6 SDMMC AHB master interface 971
24.5.7 AHB and SDMMC_CK clock relation 974
24.6 Card functional description 974
24.6.1 SD I/O mode 974
24.6.2 CMD12 send timing 982
24.6.3 Sleep (CMD5) 986
24.6.4 Interrupt mode (Wait-IRQ) 987

24.6.5 Boot operation . . . 988
24.6.6 Response R1b handling . . . 991
24.6.7 Reset and card cycle power . . . 992
24.7 Hardware flow control . . . 993
24.8 Ultra-high-speed phase I (UHS-I) voltage switch . . . 993
24.9 SDMMC interrupts . . . 996
24.10 SDMMC registers . . . 998
24.10.1 SDMMC power control register (SDMMC_POWER) . . . 998
24.10.2 SDMMC clock control register (SDMMC_CLKCR) . . . 999
24.10.3 SDMMC argument register (SDMMC_ARGR) . . . 1001
24.10.4 SDMMC command register (SDMMC_CMDR) . . . 1001
24.10.5 SDMMC command response register (SDMMC_RESPCMDR) . . . 1003
24.10.6 SDMMC response x register (SDMMC_RESPxR) . . . 1004
24.10.7 SDMMC data timer register (SDMMC_DTIMER) . . . 1004
24.10.8 SDMMC data length register (SDMMC_DLENR) . . . 1005
24.10.9 SDMMC data control register (SDMMC_DCTRL) . . . 1006
24.10.10 SDMMC data counter register (SDMMC_DCNTR) . . . 1007
24.10.11 SDMMC status register (SDMMC_STAR) . . . 1008
24.10.12 SDMMC interrupt clear register (SDMMC_ICR) . . . 1011
24.10.13 SDMMC mask register (SDMMC_MASKR) . . . 1013
24.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) . . . 1016
24.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR) . . . 1016
24.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER) . . . 1017
24.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER) . . . 1018
24.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR) . . . 1018
24.10.19 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR) . . . 1019
24.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx) . . . 1020
24.10.21 SDMMC register map . . . 1020

25 Delay block (DLYB) . . . 1023
25.1 DLYB introduction . . . 1023
25.2 DLYB main features . . . 1023
25.3 DLYB implementation . . . 1023
25.4 DLYB functional description . . . 1023
25.4.1 DLYB diagram . . . 1023

25.4.2 DLYB pins and internal signals . . . 1024
25.4.3 General description . . . 1024
25.4.4 Delay line length configuration procedure . . . 1025
25.4.5 Output clock phase configuration procedure . . . 1026
25.5 DLYB registers . . . 1026
25.5.1 DLYB control register (DLYB_CR) . . . 1026
25.5.2 DLYB configuration register (DLYB_CFGR) . . . 1027
25.5.3 DLYB register map . . . 1028

26 Analog-to-digital converters (ADC1/2) . . . 1029
26.1 ADC introduction . . . 1029
26.2 ADC main features . . . 1029
26.3 ADC implementation . . . 1031
26.4 ADC functional description . . . 1032
26.4.1 ADC block diagram . . . 1032
26.4.2 ADC pins and internal signals . . . 1033
26.4.3 ADC clocks . . . 1036
26.4.4 ADC connectivity . . . 1038
26.4.5 Slave AHB interface . . . 1040
26.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN) . . . 1040
26.4.7 Single-ended and differential input channels . . . 1041
26.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT) . . . 1041
26.4.9 ADC on-off control (ADEN, ADDIS, ADRDY) . . . 1044
26.4.10 Constraints when writing the ADC control bits . . . 1045
26.4.11 Channel selection (ADC_SQRy, ADC_JSQR) . . . 1046
26.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2) . . . 1047
26.4.13 Single conversion mode (CONT = 0) . . . 1049
26.4.14 Continuous conversion mode (CONT = 1) . . . 1049
26.4.15 Starting conversions (ADSTART, JADSTART) . . . 1050
26.4.16 ADC timing . . . 1051
26.4.17 Stopping an ongoing conversion (ADSTP, JADSTP) . . . 1052
26.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN) . . . 1053
26.4.19 Injected channel management . . . 1055
26.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . 1056
26.4.21 Queue of context for injected conversions . . . 1057

26.4.22 Programmable resolution (RES) - fast conversion mode 1065
26.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . 1066
26.4.24 End of conversion sequence (EOS, JEOS) 1066
26.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers) 1067
26.4.26 Data management 1069
26.4.27 Dynamic low-power features 1075
26.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) 1080
26.4.29 Oversampler 1084
26.4.30 Dual ADC modes 1091
26.4.31 Temperature sensor 1104
26.4.32 VBAT supply monitoring 1106
26.4.33 Monitoring the internal voltage reference 1107
26.4.34 Monitoring the supply voltage 1108
26.5 ADC interrupts 1109
26.6 ADC registers (for each ADC) 1110
26.6.1 ADC interrupt and status register (ADC_ISR) 1110
26.6.2 ADC interrupt enable register (ADC_IER) 1112
26.6.3 ADC control register (ADC_CR) 1114
26.6.4 ADC configuration register (ADC_CFGR) 1117
26.6.5 ADC configuration register 2 (ADC_CFGR2) 1122
26.6.6 ADC sample time register 1 (ADC_SMPR1) 1124
26.6.7 ADC sample time register 2 (ADC_SMPR2) 1125
26.6.8 ADC watchdog threshold register 1 (ADC_TR1) 1126
26.6.9 ADC watchdog threshold register 2 (ADC_TR2) 1127
26.6.10 ADC watchdog threshold register 3 (ADC_TR3) 1128
26.6.11 ADC regular sequence register 1 (ADC_SQR1) 1128
26.6.12 ADC regular sequence register 2 (ADC_SQR2) 1129
26.6.13 ADC regular sequence register 3 (ADC_SQR3) 1130
26.6.14 ADC regular sequence register 4 (ADC_SQR4) 1131
26.6.15 ADC regular data register (ADC_DR) 1132
26.6.16 ADC injected sequence register (ADC_JSQR) 1132
26.6.17 ADC offset y register (ADC_OFRy) 1135
26.6.18 ADC injected channel y data register (ADC_JDRy) 1136
26.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . 1137
26.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . 1137

26.6.21 ADC Differential mode selection register (ADC_DIFSEL) 1138
26.6.22 ADC calibration factors (ADC_CALFACT) 1139
26.6.23 ADC option register (ADC_OR) 1139
26.7 ADC common registers 1140
26.7.1 ADC common status register (ADC_CSR) 1140
26.7.2 ADC common control register (ADC_CCR) 1142
26.7.3 ADC common regular data register for dual mode (ADC_CDR) 1145
26.8 ADC register map 1146

27 Digital temperature sensor (DTS) 1150
27.1 DTS introduction 1150
27.2 DTS main features 1150
27.3 DTS functional description 1151
27.3.1 DTS block diagram 1151
27.3.2 DTS internal signals 1151
27.3.3 DTS block operation 1152
27.3.4 Operating modes 1152
27.3.5 Calibration 1152
27.3.6 Prescaler 1152
27.3.7 Temperature measurement principles 1153
27.3.8 Sampling time 1154
27.3.9 Quick measurement mode 1154
27.3.10 Trigger input 1155
27.3.11 On-off control and ready flag 1155
27.3.12 Temperature measurement sequence 1156
27.4 DTS low-power modes 1157
27.5 DTS interrupts 1157
27.5.1 Temperature window comparator 1157
27.5.2 Synchronous interrupt 1157
27.5.3 Asynchronous wakeup 1157
27.6 DTS registers 1158
27.6.1 Temperature sensor configuration register 1 (DTS_CFGR1) 1158
27.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1) 1160
27.6.3 Temperature sensor ramp value register (DTS_RAMPVALR) 1160
27.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1) 1161
27.6.5 Temperature sensor data register (DTS_DR) 1161

27.6.6 Temperature sensor status register (DTS_SR) 1162
27.6.7 Temperature sensor interrupt enable register (DTS_ITENR) 1163
27.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR) 1164
27.6.9 Temperature sensor option register (DTS_OR) 1165
27.6.10 DTS register map 1166

28 Digital-to-analog converter (DAC) 1167

28.1 DAC introduction 1167
28.2 DAC main features 1167
28.3 DAC implementation 1168
28.4 DAC functional description 1169
28.4.1 DAC block diagram 1169
28.4.2 DAC pins and internal signals 1170
28.4.3 DAC clocks 1171
28.4.4 DAC channel enable 1171
28.4.5 DAC data format 1171
28.4.6 DAC conversion 1173
28.4.7 DAC output voltage 1174
28.4.8 DAC trigger selection 1174
28.4.9 DMA requests 1175
28.4.10 Noise generation 1176
28.4.11 Triangle-wave generation 1177
28.4.12 DAC channel modes 1178
28.4.13 DAC channel buffer calibration 1181
28.4.14 DAC channel conversion modes 1182
28.4.15 Dual DAC channel conversion modes (if dual channels are available) 1183
28.5 DAC in low-power modes 1187
28.6 DAC interrupts 1188
28.7 DAC registers 1188
28.7.1 DAC control register (DAC_CR) 1188
28.7.2 DAC software trigger register (DAC_SWTRGR) 1191
28.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1) 1192
28.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1) 1193

28.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1) . . . 1193
28.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2) . . . 1194
28.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2) . . . 1194
28.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2) . . . 1195
28.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD) . . . 1195
28.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD) . . . 1196
28.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD) . . . 1196
28.7.12 DAC channel1 data output register (DAC_DOR1) . . . 1197
28.7.13 DAC channel2 data output register (DAC_DOR2) . . . 1197
28.7.14 DAC status register (DAC_SR) . . . 1198
28.7.15 DAC calibration control register (DAC_CCR) . . . 1199
28.7.16 DAC mode control register (DAC_MCR) . . . 1200
28.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1) . . . 1201
28.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2) . . . 1202
28.7.19 DAC sample and hold time register (DAC_SHHR) . . . 1202
28.7.20 DAC sample and hold refresh time register (DAC_SHRR) . . . 1203
28.7.21 DAC register map . . . 1204

29 Voltage reference buffer (VREFBUF) . . . 1207

29.1 VREFBUF introduction . . . 1207
29.2 VREFBUF implementation . . . 1207
29.3 VREFBUF functional description . . . 1207
29.4 VREFBUF trimming . . . 1208
29.5 VREFBUF registers . . . 1209
29.5.1 VREFBUF control and status register (VREFBUF_CSR) . . . 1209
29.5.2 VREFBUF calibration control register (VREFBUF_CCR) . . . 1210
29.5.3 VREFBUF register map . . . 1210

30 Digital camera interface (DCMI) . . . 1211

30.1 DCMI introduction . . . 1211

30.2 DCMI main features 1211
30.3 DCMI functional description 1211
30.3.1 DCMI block diagram 1212
30.3.2 DCMI pins and internal signals 1212
30.3.3 DCMI clocks 1213
30.3.4 DCMI DMA interface 1213
30.3.5 DCMI physical interface 1213
30.3.6 DCMI synchronization 1215
30.3.7 DCMI capture modes 1217
30.3.8 DCMI crop feature 1218
30.3.9 DCMI JPEG format 1219
30.3.10 DCMI FIFO 1219
30.3.11 DCMI data format description 1220
30.4 DCMI interrupts 1222
30.5 DCMI registers 1222
30.5.1 DCMI control register (DCMI_CR) 1222
30.5.2 DCMI status register (DCMI_SR) 1225
30.5.3 DCMI raw interrupt status register (DCMI_RIS) 1225
30.5.4 DCMI interrupt enable register (DCMI_IER) 1226
30.5.5 DCMI masked interrupt status register (DCMI_MIS) 1227
30.5.6 DCMI interrupt clear register (DCMI_ICR) 1228
30.5.7 DCMI embedded synchronization code register (DCMI_ESCR) 1229
30.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR) . 1229
30.5.9 DCMI crop window start (DCMI_CWSTRT) 1230
30.5.10 DCMI crop window size (DCMI_CWSIZE) 1231
30.5.11 DCMI data register (DCMI_DR) 1231
30.5.12 DCMI register map 1232

31 Parallel synchronous slave interface (PSSI) 1233
31.1 PSSI introduction 1233
31.2 PSSI main features 1233
31.3 PSSI functional description 1233
31.3.1 PSSI block diagram 1234
31.3.2 PSSI pins and internal signals 1234
31.3.3 PSSI clock 1235
31.3.4 PSSI data management 1235

31.3.5 PSSI optional control signals . . . 1237
31.4 PSSI interrupts . . . 1240
31.5 PSSI registers . . . 1241
31.5.1 PSSI control register (PSSI_CR) . . . 1241
31.5.2 PSSI status register (PSSI_SR) . . . 1242
31.5.3 PSSI raw interrupt status register (PSSI_RIS) . . . 1243
31.5.4 PSSI interrupt enable register (PSSI_IER) . . . 1244
31.5.5 PSSI masked interrupt status register (PSSI_MIS) . . . 1244
31.5.6 PSSI interrupt clear register (PSSI_ICR) . . . 1245
31.5.7 PSSI data register (PSSI_DR) . . . 1245
31.5.8 PSSI register map . . . 1246

32 True random number generator (RNG) . . . 1247
32.1 RNG introduction . . . 1247
32.2 RNG main features . . . 1247
32.3 RNG functional description . . . 1248
32.3.1 RNG block diagram . . . 1248
32.3.2 RNG internal signals . . . 1248
32.3.3 Random number generation . . . 1248
32.3.4 RNG initialization . . . 1251
32.3.5 RNG operation . . . 1252
32.3.6 RNG clocking . . . 1254
32.3.7 Error management . . . 1254
32.3.8 RNG low-power use . . . 1255
32.4 RNG interrupts . . . 1256
32.5 RNG processing time . . . 1256
32.6 RNG entropy source validation . . . 1257
32.6.1 Introduction . . . 1257
32.6.2 Validation conditions . . . 1257
32.7 RNG registers . . . 1258
32.7.1 RNG control register (RNG_CR) . . . 1258
32.7.2 RNG status register (RNG_SR) . . . 1260
32.7.3 RNG data register (RNG_DR) . . . 1261
32.7.4 RNG noise source control register (RNG_NSCR) . . . 1262
32.7.5 RNG health test control register (RNG_HTCR) . . . 1263
32.7.6 RNG register map . . . 1263

33 AES hardware accelerator (AES) 1264
33.1 AES introduction 1264
33.2 AES main features 1264
33.3 AES implementation 1265
33.4 AES functional description 1265
33.4.1 AES block diagram 1265
33.4.2 AES internal signals 1266
33.4.3 AES reset and clocks 1266
33.4.4 AES symmetric cipher implementation 1266
33.4.5 AES encryption or decryption typical usage 1267
33.4.6 AES authenticated encryption, decryption, and cipher-based message authentication 1270
33.4.7 AES ciphertext stealing and data padding 1270
33.4.8 AES suspend and resume operations 1271
33.4.9 AES basic chaining modes (ECB, CBC) 1271
33.4.10 AES counter (CTR) mode 1275
33.4.11 AES Galois/counter mode (GCM) 1277
33.4.12 AES Galois message authentication code (GMAC) 1282
33.4.13 AES counter with CBC-MAC (CCM) 1283
33.4.14 AES key sharing with secure AES co-processor 1288
33.4.15 AES data registers and data swapping 1289
33.4.16 AES key registers 1291
33.4.17 AES initialization vector registers 1291
33.4.18 AES error management 1292
33.5 AES interrupts 1293
33.6 AES DMA requests 1293
33.7 AES processing latency 1294
33.8 AES registers 1296
33.8.1 AES control register (AES_CR) 1296
33.8.2 AES status register (AES_SR) 1298
33.8.3 AES data input register (AES_DINR) 1299
33.8.4 AES data output register (AES_DOUTR) 1300
33.8.5 AES key register 0 (AES_KEYR0) 1300
33.8.6 AES key register 1 (AES_KEYR1) 1301
33.8.7 AES key register 2 (AES_KEYR2) 1301
33.8.8 AES key register 3 (AES_KEYR3) 1301

33.8.9 AES initialization vector register 0 (AES_IVR0) 1302
33.8.10 AES initialization vector register 1 (AES_IVR1) 1302
33.8.11 AES initialization vector register 2 (AES_IVR2) 1302
33.8.12 AES initialization vector register 3 (AES_IVR3) 1303
33.8.13 AES key register 4 (AES_KEYR4) 1303
33.8.14 AES key register 5 (AES_KEYR5) 1303
33.8.15 AES key register 6 (AES_KEYR6) 1304
33.8.16 AES key register 7 (AES_KEYR7) 1304
33.8.17 AES suspend registers (AES_SUSPRx) 1304
33.8.18 AES interrupt enable register (AES_IER) 1305
33.8.19 AES interrupt status register (AES_ISR) 1306
33.8.20 AES interrupt clear register (AES_ICR) 1307
33.8.21 AES register map 1307

34 Secure AES coprocessor (SAES) 1310

34.1 SAES introduction 1310
34.2 SAES main features 1310
34.3 SAES implementation 1311
34.4 SAES functional description 1311
34.4.1 SAES block diagram 1311
34.4.2 SAES internal signals 1312
34.4.3 SAES reset and clocks 1313
34.4.4 SAES symmetric cipher implementation 1313
34.4.5 SAES encryption or decryption typical usage 1314
34.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication 1316
34.4.7 SAES ciphertext stealing and data padding 1317
34.4.8 SAES suspend and resume operations 1317
34.4.9 SAES basic chaining modes (ECB, CBC) 1318
34.4.10 SAES counter (CTR) mode 1322
34.4.11 SAES Galois/counter mode (GCM) 1324
34.4.12 SAES Galois message authentication code (GMAC) 1328
34.4.13 SAES counter with CBC-MAC (CCM) 1330
34.4.14 SAES operation with wrapped keys 1335
34.4.15 SAES operation with shared keys 1339
34.4.16 SAES data registers and data swapping 1340
34.4.17 SAES key registers 1343

34.4.18 SAES initialization vector registers 1344
34.4.19 SAES error management 1345
34.5 SAES interrupts 1347
34.6 SAES DMA requests 1347
34.7 SAES processing latency 1348
34.8 SAES registers 1350
34.8.1 SAES control register (SAES_CR) 1350
34.8.2 SAES status register (SAES_SR) 1353
34.8.3 SAES data input register (SAES_DINR) 1354
34.8.4 SAES data output register (SAES_DOUTR) 1355
34.8.5 SAES key register 0 (SAES_KEYR0) 1355
34.8.6 SAES key register 1 (SAES_KEYR1) 1356
34.8.7 SAES key register 2 (SAES_KEYR2) 1356
34.8.8 SAES key register 3 (SAES_KEYR3) 1356
34.8.9 SAES initialization vector register 0 (SAES_IVR0) 1357
34.8.10 SAES initialization vector register 1 (SAES_IVR1) 1357
34.8.11 SAES initialization vector register 2 (SAES_IVR2) 1357
34.8.12 SAES initialization vector register 3 (SAES_IVR3) 1358
34.8.13 SAES key register 4 (SAES_KEYR4) 1358
34.8.14 SAES key register 5 (SAES_KEYR5) 1358
34.8.15 SAES key register 6 (SAES_KEYR6) 1359
34.8.16 SAES key register 7 (SAES_KEYR7) 1359
34.8.17 SAES suspend registers (SAES_SUSPRx) 1359
34.8.18 SAES interrupt enable register (SAES_IER) 1360
34.8.19 SAES interrupt status register (SAES_ISR) 1361
34.8.20 SAES interrupt clear register (SAES_ICR) 1362
34.8.21 SAES register map 1363

35 Hash processor (HASH) 1365
35.1 HASH introduction 1365
35.2 HASH main features 1365
35.3 HASH implementation 1366
35.4 HASH functional description 1366
35.4.1 HASH block diagram 1366
35.4.2 HASH internal signals 1366
35.4.3 About secure hash algorithms 1367

35.4.4 Message data feeding . . . 1367
35.4.5 Message digest computing . . . 1368
35.4.6 Message padding . . . 1370
35.4.7 HMAC operation . . . 1371
35.4.8 HASH suspend/resume operations . . . 1373
35.4.9 HASH DMA interface . . . 1375
35.4.10 HASH error management . . . 1376
35.4.11 HASH processing time . . . 1376
35.5 HASH interrupts . . . 1377
35.6 HASH registers . . . 1377
35.6.1 HASH control register (HASH_CR) . . . 1377
35.6.2 HASH data input register (HASH_DIN) . . . 1379
35.6.3 HASH start register (HASH_STR) . . . 1380
35.6.4 HASH digest registers . . . 1381
35.6.5 HASH interrupt enable register (HASH_IMR) . . . 1383
35.6.6 HASH status register (HASH_SR) . . . 1383
35.6.7 HASH context swap registers . . . 1384
35.6.8 HASH register map . . . 1385

36 Public key accelerator (PKA) . . . 1387
36.1 PKA introduction . . . 1387
36.2 PKA main features . . . 1387
36.3 PKA implementation . . . 1387
36.4 PKA functional description . . . 1388
36.4.1 PKA block diagram . . . 1388
36.4.2 PKA internal signals . . . 1388
36.4.3 PKA reset and clocks . . . 1388
36.4.4 PKA public key acceleration . . . 1389
36.4.5 Typical applications for PKA . . . 1391
36.4.6 PKA procedure to perform an operation . . . 1393
36.4.7 PKA error management . . . 1394
36.5 PKA operating modes . . . 1395
36.5.1 Introduction . . . 1395
36.5.2 Montgomery parameter computation . . . 1396
36.5.3 Modular addition . . . 1396
36.5.4 Modular subtraction . . . 1397

36.5.5 Modular and Montgomery multiplication 1397
36.5.6 Modular exponentiation 1398
36.5.7 Modular inversion 1400
36.5.8 Modular reduction 1400
36.5.9 Arithmetic addition 1401
36.5.10 Arithmetic subtraction 1401
36.5.11 Arithmetic multiplication 1402
36.5.12 Arithmetic comparison 1402
36.5.13 RSA CRT exponentiation 1402
36.5.14 Point on elliptic curve Fp check 1403
36.5.15 ECC Fp scalar multiplication 1404
36.5.16 ECDSA sign 1405
36.5.17 ECDSA verification 1407
36.5.18 ECC complete addition 1408
36.5.19 ECC double base ladder 1408
36.5.20 ECC projective to affine 1409
36.6 Example of configurations and processing times 1410
36.6.1 Supported elliptic curves 1410
36.6.2 Computation times 1412
36.7 PKA interrupts 1414
36.8 PKA registers 1415
36.8.1 PKA control register (PKA_CR) 1415
36.8.2 PKA status register (PKA_SR) 1417
36.8.3 PKA clear flag register (PKA_CLRFR) 1418
36.8.4 PKA RAM 1418
36.8.5 PKA register map 1419

37 On-the-fly decryption engine (OTFDEC) 1420
37.1 OTFDEC introduction 1420
37.2 OTFDEC main features 1420
37.3 OTFDEC functional description 1421
37.3.1 OTFDEC block diagram 1421
37.3.2 OTFDEC internal signals 1421
37.3.3 OTFDEC on-the-fly decryption 1422
37.3.4 OTFDEC usage of AES in counter mode decryption 1423
37.3.5 Flow control management 1424

37.3.6 OTFDEC error management 1424
37.4 OTFDEC interrupts 1425
37.5 OTFDEC application information 1425
37.5.1 OTFDEC initialization process 1425
37.5.2 OTFDEC and power management 1427
37.5.3 Encrypting for OTFDEC 1427
37.5.4 OTFDEC key CRC source code 1428
37.6 OTFDEC registers 1429
37.6.1 OTFDEC control register (OTFDEC_CR) 1429
37.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR) 1430
37.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR) 1430
37.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR) 1432
37.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR) . 1432
37.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0) 1433
37.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1) 1434
37.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0) 1434
37.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1) 1435
37.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2) 1435
37.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3) 1436
37.6.12 OTFDEC interrupt status register (OTFDEC_ISR) 1436
37.6.13 OTFDEC interrupt clear register (OTFDEC_ICR) 1437
37.6.14 OTFDEC interrupt enable register (OTFDEC_IER) 1438
37.6.15 OTFDEC register map 1439

38 Advanced-control timers (TIM1/TIM8) 1443
38.1 TIM1/TIM8 introduction 1443
38.2 TIM1/TIM8 main features 1443
38.3 TIM1/TIM8 functional description 1444
38.3.1 Block diagram 1444
38.3.2 TIM1/TIM8 pins and internal signals 1445
38.3.3 Time-base unit 1449
38.3.4 Counter modes 1451
38.3.5 Repetition counter 1463
38.3.6 External trigger input 1464
38.3.7 Clock selection 1465

38.3.8 Capture/compare channels . . . 1469
38.3.9 Input capture mode . . . 1472
38.3.10 PWM input mode . . . 1473
38.3.11 Forced output mode . . . 1474
38.3.12 Output compare mode . . . 1474
38.3.13 PWM mode . . . 1476
38.3.14 Asymmetric PWM mode . . . 1484
38.3.15 Combined PWM mode . . . 1485
38.3.16 Combined 3-phase PWM mode . . . 1486
38.3.17 Complementary outputs and dead-time insertion . . . 1487
38.3.18 Using the break function . . . 1490
38.3.19 Bidirectional break inputs . . . 1496
38.3.20 Clearing the tim_ocxref signal on an external event . . . 1497
38.3.21 6-step PWM generation . . . 1499
38.3.22 One-pulse mode . . . 1500
38.3.23 Retriggerable One-pulse mode . . . 1502
38.3.24 Pulse on compare mode . . . 1503
38.3.25 Encoder interface mode . . . 1505
38.3.26 Direction bit output . . . 1522
38.3.27 UIF bit remapping . . . 1523
38.3.28 Timer input XOR function . . . 1523
38.3.29 Interfacing with Hall sensors . . . 1523
38.3.30 Timer synchronization . . . 1525
38.3.31 ADC triggers . . . 1530
38.3.32 DMA burst mode . . . 1530
38.3.33 TIM1/TIM8 DMA requests . . . 1531
38.3.34 Debug mode . . . 1531
38.4 TIM1/TIM8 low-power modes . . . 1532
38.5 TIM1/TIM8 interrupts . . . 1532
38.6 TIM1/TIM8 registers . . . 1533
38.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8) . . . 1533
38.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8) . . . 1534
38.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8) . . . 1538
38.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8) . . . 1542
38.6.5 TIMx status register (TIMx_SR)(x = 1, 8) . . . 1543
38.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8) . . . 1546

38.6.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 1, 8) 1547
38.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8) 1549
38.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8) 1552
38.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8) 1553
38.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8) 1556
38.6.12 TIMx counter (TIMx_CNT)(x = 1, 8) 1560
38.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8) 1560
38.6.14 TIMx autoreload register (TIMx_ARR)(x = 1, 8) 1561
38.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8) 1561
38.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8) 1562
38.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8) 1562
38.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8) 1563
38.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8) 1564
38.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8) 1565
38.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8) 1569
38.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8) 1570
38.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8) 1571
38.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8) 1572
38.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8) 1573
38.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8) 1574
38.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8) 1575
38.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8) 1578
38.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8) 1580
38.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8) 1582
38.6.31 TIMx register map 1582

39 General-purpose timers (TIM2/TIM3/TIM4/TIM5) 1585

39.1 TIM2/TIM3/TIM4/TIM5 introduction 1585
39.2 TIM2/TIM3/TIM4/TIM5 main features 1585
39.3 TIM2/TIM3/TIM4/TIM5 implementation 1586
39.4 TIM2/TIM3/TIM4/TIM5 functional description 1587
39.4.1 Block diagram 1587
39.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals 1588

39.4.3 Time-base unit . . . 1591
39.4.4 Counter modes . . . 1593
39.4.5 Clock selection . . . 1605
39.4.6 Capture/compare channels . . . 1609
39.4.7 Input capture mode . . . 1611
39.4.8 PWM input mode . . . 1612
39.4.9 Forced output mode . . . 1613
39.4.10 Output compare mode . . . 1613
39.4.11 PWM mode . . . 1615
39.4.12 Asymmetric PWM mode . . . 1623
39.4.13 Combined PWM mode . . . 1624
39.4.14 Clearing the tim_ocxref signal on an external event . . . 1625
39.4.15 One-pulse mode . . . 1627
39.4.16 Retriggerable one-pulse mode . . . 1628
39.4.17 Pulse on compare mode . . . 1629
39.4.18 Encoder interface mode . . . 1631
39.4.19 Direction bit output . . . 1649
39.4.20 UIF bit remapping . . . 1650
39.4.21 Timer input XOR function . . . 1650
39.4.22 Timers and external trigger synchronization . . . 1650
39.4.23 Timer synchronization . . . 1654
39.4.24 ADC triggers . . . 1659
39.4.25 DMA burst mode . . . 1660
39.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests . . . 1661
39.4.27 Debug mode . . . 1661
39.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes . . . 1661
39.4.29 TIM2/TIM3/TIM4/TIM5 interrupts . . . 1662
39.5 TIM2/TIM3/TIM4/TIM5 registers . . . 1663
39.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5) . . . 1663
39.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5) . . . 1664
39.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5) . . . 1666
39.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5) . . . 1670
39.5.5 TIMx status register (TIMx_SR)(x = 2 to 5) . . . 1671
39.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5) . . . 1673
39.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5) . 1674
39.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5) . . . 1676

39.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5) . 1678
39.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5) 1679
39.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5) . . . 1682
39.5.12 TIMx counter (TIMx_CNT)(x = 3, 4) 1683
39.5.13 TIMx counter (TIMx_CNT)(x = 2, 5) 1684
39.5.14 TIMx prescaler (TIMx_PSC)(x = 2 to 5) 1684
39.5.15 TIMx autoreload register (TIMx_ARR)(x = 3, 4) 1685
39.5.16 TIMx autoreload register (TIMx_ARR)(x = 2, 5) 1685
39.5.17 TIMx capture/compare register 1 (TIMx_CCR1)(x = 3, 4) 1686
39.5.18 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2, 5) 1687
39.5.19 TIMx capture/compare register 2 (TIMx_CCR2)(x = 3, 4) 1687
39.5.20 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2, 5) 1688
39.5.21 TIMx capture/compare register 3 (TIMx_CCR3)(x = 3, 4) 1689
39.5.22 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2, 5) 1690
39.5.23 TIMx capture/compare register 4 (TIMx_CCR4)(x = 3, 4) 1691
39.5.24 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2, 5) 1692
39.5.25 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5) 1693
39.5.26 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5) 1694
39.5.27 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5) 1695
39.5.28 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5) 1696
39.5.29 TIMx DMA control register (TIMx_DCR)(x = 2 to 5) 1697
39.5.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5) 1698
39.5.31 TIMx register map 1699

40 Basic timers (TIM6/TIM7) 1702

40.1 TIM6/TIM7 introduction 1702
40.2 TIM6/TIM7 main features 1702
40.3 TIM6/TIM7 functional description 1703
40.3.1 TIM6/TIM7 block diagram 1703
40.3.2 TIM6/TIM7 internal signals 1703
40.3.3 TIM6/TIM7 clocks 1704
40.3.4 Time-base unit 1704
40.3.5 Counting mode 1706
40.3.6 UIF bit remapping 1713
40.3.7 ADC triggers 1714
40.3.8 TIM6/TIM7 DMA requests 1714

40.3.9 Debug mode . . . 1714
40.3.10 TIM6/TIM7 low-power modes . . . 1714
40.3.11 TIM6/TIM7 interrupts . . . 1714
40.4 TIM6/TIM7 registers . . . 1715
40.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) . . . 1715
40.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) . . . 1717
40.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) . . . 1717
40.4.4 TIMx status register (TIMx_SR)(x = 6 to 7) . . . 1718
40.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7) . . . 1718
40.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7) . . . 1718
40.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7) . . . 1719
40.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7) . . . 1719
40.4.9 TIMx register map . . . 1720

41 General-purpose timers (TIM12/TIM13/TIM14) . . . 1721
41.1 TIM12/TIM13/TIM14 introduction . . . 1721
41.2 TIM12 main features . . . 1721
41.3 TIM13/TIM14 main features . . . 1722
41.4 TIM12/TIM13/TIM14 functional description . . . 1723
41.4.1 Block diagram . . . 1723
41.4.2 TIM12/TIM13/TIM14 pins and internal signals . . . 1724
41.4.3 Time-base unit . . . 1727
41.4.4 Counter modes . . . 1729
41.4.5 Clock selection . . . 1732
41.4.6 Capture/compare channels . . . 1734
41.4.7 Input capture mode . . . 1736
41.4.8 PWM input mode (TIM12 only) . . . 1737
41.4.9 Forced output mode . . . 1738
41.4.10 Output compare mode . . . 1739
41.4.11 PWM mode . . . 1740
41.4.12 Combined PWM mode (TIM12 only) . . . 1745
41.4.13 One-pulse mode . . . 1746
41.4.14 Retriggerable one pulse mode (TIM12 only) . . . 1748
41.4.15 UIF bit remapping . . . 1749
41.4.16 Timer input XOR function . . . 1749
41.4.17 TIM12 external trigger synchronization . . . 1749
41.4.18 Slave mode – combined reset + trigger mode . . . 1752

41.4.19 Slave mode – combined reset + gated mode 1752
41.4.20 Timer synchronization (TIM12 only) 1752
41.4.21 Using timer output as trigger for other timers (TIM13/TIM14 only) 1752
41.4.22 ADC triggers (TIM12 only) 1752
41.4.23 Debug mode 1752
41.5 TIM12/TIM13/TIM14 low-power modes 1753
41.6 TIM12/TIM13/TIM14 interrupts 1753
41.7 TIM12 registers 1753
41.7.1 TIM12 control register 1 (TIM12_CR1) 1753
41.7.2 TIM12 control register 2 (TIM12_CR2) 1755
41.7.3 TIM12 slave mode control register (TIM12_SMCR) 1755
41.7.4 TIM12 interrupt enable register (TIM12_DIER) 1758
41.7.5 TIM12 status register (TIM12_SR) 1759
41.7.6 TIM12 event generation register (TIM12_EGR) 1760
41.7.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1) 1761
41.7.8 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1) 1762
41.7.9 TIM12 capture/compare enable register (TIM12_CCER) 1765
41.7.10 TIM12 counter (TIM12_CNT) 1766
41.7.11 TIM12 prescaler (TIM12_PSC) 1767
41.7.12 TIM12 autoreload register (TIM12_ARR) 1767
41.7.13 TIM12 capture/compare register 1 (TIM12_CCR1) 1768
41.7.14 TIM12 capture/compare register 2 (TIM12_CCR2) 1768
41.7.15 TIM12 timer input selection register (TIM12_TISEL) 1769
41.7.16 TIM12 register map 1770
41.8 TIM13/TIM14 registers 1772
41.8.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14) 1772
41.8.2 TIMx interrupt enable register (TIMx_DIER)(x = 13, 14) 1773
41.8.3 TIMx status register (TIMx_SR)(x = 13, 14) 1773
41.8.4 TIMx event generation register (TIMx_EGR)(x = 13, 14) 1774
41.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13, 14) 1775
41.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13, 14) 1776
41.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 13, 14) 1778
41.8.8 TIMx counter (TIMx_CNT)(x = 13, 14) 1779

41.8.9 TIMx prescaler (TIMx_PSC)(x = 13, 14) 1780
41.8.10 TIMx autoreload register (TIMx_ARR)(x = 13, 14) 1780
41.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13, 14) 1781
41.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 13, 14) 1781
41.8.13 TIM13/TIM14 register map 1782

42 General purpose timers (TIM15/TIM16/TIM17) 1784

42.1 TIM15/TIM16/TIM17 introduction 1784
42.2 TIM15 main features 1784
42.3 TIM16/TIM17 main features 1785
42.4 TIM15/TIM16/TIM17 functional description 1786
42.4.1 Block diagram 1786
42.4.2 TIM15/TIM16/TIM17 pins and internal signals 1787
42.4.3 Time-base unit 1790
42.4.4 Counter modes 1792
42.4.5 Repetition counter 1796
42.4.6 Clock selection 1797
42.4.7 Capture/compare channels 1799
42.4.8 Input capture mode 1801
42.4.9 PWM input mode (only for TIM15) 1803
42.4.10 Forced output mode 1804
42.4.11 Output compare mode 1804
42.4.12 PWM mode 1806
42.4.13 Combined PWM mode (TIM15 only) 1811
42.4.14 Complementary outputs and dead-time insertion 1812
42.4.15 Using the break function 1815
42.4.16 Bidirectional break input 1819
42.4.17 Clearing the tim_ocxref signal on an external event 1820
42.4.18 6-step PWM generation 1821
42.4.19 One-pulse mode 1823
42.4.20 Retriggerable one pulse mode (TIM15 only) 1824
42.4.21 UIF bit remapping 1825
42.4.22 Timer input XOR function (TIM15 only) 1825
42.4.23 External trigger synchronization (TIM15 only) 1825
42.4.24 Slave mode – combined reset + trigger mode (TIM15 only) 1828
42.4.25 Slave mode – combined reset + gated mode (TIM15 only) 1828
42.4.26 Timer synchronization (TIM15 only) 1829

42.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 1829
42.4.28 ADC triggers (TIM15 only) 1829
42.4.29 DMA burst mode 1829
42.4.30 TIM15/TIM16/TIM17 DMA requests 1830
42.4.31 Debug mode 1830
42.5 TIM15/TIM16/TIM17 low-power modes 1831
42.6 TIM15/TIM16/TIM17 interrupts 1831
42.7 TIM15 registers 1832
42.7.1 TIM15 control register 1 (TIM15_CR1) 1832
42.7.2 TIM15 control register 2 (TIM15_CR2) 1833
42.7.3 TIM15 slave mode control register (TIM15_SMCR) 1835
42.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER) 1837
42.7.5 TIM15 status register (TIM15_SR) 1838
42.7.6 TIM15 event generation register (TIM15_EGR) 1840
42.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) 1841
42.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 1842
42.7.9 TIM15 capture/compare enable register (TIM15_CCER) 1845
42.7.10 TIM15 counter (TIM15_CNT) 1848
42.7.11 TIM15 prescaler (TIM15_PSC) 1848
42.7.12 TIM15 autoreload register (TIM15_ARR) 1849
42.7.13 TIM15 repetition counter register (TIM15_RCR) 1849
42.7.14 TIM15 capture/compare register 1 (TIM15_CCR1) 1850
42.7.15 TIM15 capture/compare register 2 (TIM15_CCR2) 1851
42.7.16 TIM15 break and dead-time register (TIM15_BDTR) 1851
42.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2) 1854
42.7.18 TIM15 input selection register (TIM15_TISEL) 1855
42.7.19 TIM15 alternate function register 1 (TIM15_AF1) 1856
42.7.20 TIM15 alternate function register 2 (TIM15_AF2) 1858
42.7.21 TIM15 DMA control register (TIM15_DCR) 1859
42.7.22 TIM15 DMA address for full transfer (TIM15_DMAR) 1860
42.7.23 TIM15 register map 1860
42.8 TIM16/TIM17 registers 1863
42.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) 1863
42.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) 1864
42.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) 1865
42.8.4 TIMx status register (TIMx_SR)(x = 16 to 17) 1866

42.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) 1867
42.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17) 1868
42.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) 1869
42.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . 1871
42.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17) 1874
42.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) 1874
42.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17) 1875
42.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) 1875
42.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) 1876
42.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) 1877
42.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17) 1880
42.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17) 1881
42.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17) 1881
42.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17) 1884
42.8.19 TIM17 option register 1 (TIM17_OR1) 1884
42.8.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) 1885
42.8.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) 1886
42.8.22 TIM16/TIM17 register map 1887

43 Low-power timer (LPTIM) 1890

43.1 LPTIM introduction 1890
43.2 LPTIM main features 1890
43.3 LPTIM implementation 1890
43.4 LPTIM functional description 1892
43.4.1 LPTIM block diagram 1892
43.4.2 LPTIM pins and internal signals 1893
43.4.3 LPTIM input and trigger mapping 1895
43.4.4 LPTIM reset and clocks 1896
43.4.5 Glitch filter 1897
43.4.6 Prescaler 1898
43.4.7 Trigger multiplexer 1898
43.4.8 Operating mode 1899
43.4.9 Timeout function 1901
43.4.10 Waveform generation 1901
43.4.11 Register update 1902

43.4.12 Counter mode . . . 1903
43.4.13 Timer enable . . . 1903
43.4.14 Timer counter reset . . . 1904
43.4.15 Encoder mode . . . 1904
43.4.16 Repetition counter . . . 1906
43.4.17 Capture/compare channels . . . 1907
43.4.18 Input capture mode . . . 1908
43.4.19 PWM mode . . . 1910
43.4.20 DMA requests . . . 1912
43.4.21 Debug mode . . . 1913
43.5 LPTIM low-power modes . . . 1913
43.6 LPTIM interrupts . . . 1913
43.7 LPTIM registers . . . 1914
43.7.1 LPTIM4 interrupt and status register (LPTIM4_ISR) . . . 1915
43.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3, 5, 6) . . . 1916
43.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3, 5, 6) . . . 1918
43.7.4 LPTIM4 interrupt clear register (LPTIM4_ICR) . . . 1920
43.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3, 5, 6) . . . 1921
43.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3, 5, 6) . . . 1922
43.7.7 LPTIM4 interrupt enable register (LPTIM4_DIER) . . . 1923
43.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3, 5, 6) . . . 1925
43.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3, 5, 6) . . . 1926
43.7.10 LPTIM configuration register (LPTIM_CFGR) . . . 1928
43.7.11 LPTIM control register (LPTIM_CR) . . . 1931
43.7.12 LPTIM compare register 1 (LPTIM_CCR1) . . . 1932
43.7.13 LPTIM autoreload register (LPTIM_ARR) . . . 1933
43.7.14 LPTIM counter register (LPTIM_CNT) . . . 1933
43.7.15 LPTIM configuration register 2 (LPTIM_CFGR2) . . . 1934
43.7.16 LPTIM repetition register (LPTIM_RCR) . . . 1935
43.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) . . . 1935
43.7.18 LPTIM compare register 2 (LPTIM_CCR2) . . . 1938
43.7.19 LPTIM register map . . . 1938

44 Independent watchdog (IWDG) 1941
44.1 IWDG introduction 1941
44.2 IWDG main features 1941
44.3 IWDG implementation 1941
44.4 IWDG functional description 1942
44.4.1 IWDG block diagram 1942
44.4.2 IWDG internal signals 1943
44.4.3 Software and hardware watchdog modes 1943
44.4.4 Window option 1944
44.4.5 Debug 1947
44.4.6 Register access protection 1947
44.5 IWDG low power modes 1948
44.6 IWDG interrupts 1948
44.7 IWDG registers 1950
44.7.1 IWDG key register (IWDG_KR) 1951
44.7.2 IWDG prescaler register (IWDG_PR) 1951
44.7.3 IWDG reload register (IWDG_RLR) 1952
44.7.4 IWDG status register (IWDG_SR) 1952
44.7.5 IWDG window register (IWDG_WINR) 1954
44.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 1954
44.7.7 IWDG register map 1956

45 System window watchdog (WWDG) 1957
45.1 WWDG introduction 1957
45.2 WWDG main features 1957
45.3 WWDG implementation 1957
45.4 WWDG functional description 1958
45.4.1 WWDG block diagram 1958
45.4.2 WWDG internal signals 1958
45.4.3 Enabling the watchdog 1959
45.4.4 Controlling the down-counter 1959
45.4.5 How to program the watchdog timeout 1959
45.4.6 Debug mode 1960
45.5 WWDG interrupts 1961
45.6 WWDG registers 1961

45.6.1 WWDG control register (WWDG_CR) 1961
45.6.2 WWDG configuration register (WWDG_CFR) 1962
45.6.3 WWDG status register (WWDG_SR) 1963
45.6.4 WWDG register map 1963

46 Real-time clock (RTC) 1964
46.1 RTC ntroduction 1964
46.2 RTC main features 1964
46.3 RTC functional description 1965
46.3.1 RTC block diagram 1965
46.3.2 RTC pins and internal signals 1967
46.3.3 GPIOs controlled by the RTC and TAMP 1968
46.3.4 RTC secure protection modes 1972
46.3.5 RTC privilege protection modes 1974
46.3.6 Clock and prescalers 1975
46.3.7 Real-time clock and calendar 1976
46.3.8 Calendar ultra-low power mode 1976
46.3.9 Programmable alarms 1977
46.3.10 Periodic auto-wake-up 1977
46.3.11 RTC initialization and configuration 1978
46.3.12 Reading the calendar 1980
46.3.13 Resetting the RTC 1981
46.3.14 RTC synchronization 1982
46.3.15 RTC reference clock detection 1982
46.3.16 RTC smooth digital calibration 1983
46.3.17 Timestamp function 1985
46.3.18 Calibration clock output 1986
46.3.19 Tamper and alarm output 1986
46.4 RTC low-power modes 1987
46.5 RTC interrupts 1987
46.6 RTC registers 1989
46.6.1 RTC time register (RTC_TR) 1989
46.6.2 RTC date register (RTC_DR) 1990
46.6.3 RTC subsecond register (RTC_SSR) 1991
46.6.4 RTC initialization control and status register (RTC_ICSR) 1992
46.6.5 RTC prescaler register (RTC_PRER) 1994

46.6.6 RTC wake-up timer register (RTC_WUTR) 1995
46.6.7 RTC control register (RTC_CR) 1995
46.6.8 RTC privilege mode control register (RTC_PRIVCFGR) 1999
46.6.9 RTC secure configuration register (RTC_SECCFGR) 2001
46.6.10 RTC write protection register (RTC_WPR) 2002
46.6.11 RTC calibration register (RTC_CALR) 2003
46.6.12 RTC shift control register (RTC_SHIFTR) 2004
46.6.13 RTC timestamp time register (RTC_TSTR) 2005
46.6.14 RTC timestamp date register (RTC_TSDR) 2006
46.6.15 RTC timestamp subsecond register (RTC_TSSSR) 2007
46.6.16 RTC alarm A register (RTC_ALRMAR) 2007
46.6.17 RTC alarm A subsecond register (RTC_ALRMASSR) 2009
46.6.18 RTC alarm B register (RTC_ALRMBR) 2010
46.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR) 2011
46.6.20 RTC status register (RTC_SR) 2012
46.6.21 RTC nonsecure masked interrupt status register (RTC_MISR) 2013
46.6.22 RTC secure masked interrupt status register (RTC_SMISR) 2014
46.6.23 RTC status clear register (RTC_SCR) 2015
46.6.24 RTC option register (RTC_OR) 2016
46.6.25 RTC alarm A binary mode register (RTC_ALRABINR) 2017
46.6.26 RTC alarm B binary mode register (RTC_ALRBBINR) 2017
46.6.27 RTC register map 2019

47 Tamper and backup registers (TAMP) 2022

47.1 TAMP introduction 2022
47.2 TAMP main features 2023
47.3 TAMP functional description 2024
47.3.1 TAMP block diagram 2024
47.3.2 TAMP pins and internal signals 2025
47.3.3 GPIOs controlled by the RTC and TAMP 2028
47.3.4 TAMP register write protection 2028
47.3.5 TAMP secure protection modes 2028
47.3.6 Backup registers protection zones 2029
47.3.7 TAMP privilege protection modes 2029
47.3.8 Boot hardware key (BHK) 2030
47.3.9 Tamper detection 2030
47.3.10 TAMP backup registers and other device secrets erase 2030

47.3.11 Tamper detection configuration and initialization 2032
47.4 TAMP low-power modes 2038
47.5 TAMP interrupts 2039
47.6 TAMP registers 2039
47.6.1 TAMP control register 1 (TAMP_CR1) 2039
47.6.2 TAMP control register 2 (TAMP_CR2) 2041
47.6.3 TAMP control register 3 (TAMP_CR3) 2044
47.6.4 TAMP filter control register (TAMP_FLTCR) 2045
47.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) 2047
47.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) 2049
47.6.7 TAMP active tamper output register (TAMP_ATOR) 2050
47.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) 2051
47.6.9 TAMP secure configuration register (TAMP_SECCFGR) 2054
47.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) 2056
47.6.11 TAMP interrupt enable register (TAMP_IER) 2057
47.6.12 TAMP status register (TAMP_SR) 2059
47.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR) . . 2061
47.6.14 TAMP secure masked interrupt status register (TAMP_SMISR) 2062
47.6.15 TAMP status clear register (TAMP_SCR) 2064
47.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R) 2066
47.6.17 TAMP option register (TAMP_OR) 2066
47.6.18 TAMP resources protection configuration register (TAMP_RPCFGR) 2067
47.6.19 TAMP backup x register (TAMP_BKPxR) 2068
47.6.20 TAMP register map 2069

48 Inter-integrated circuit interface (I2C) 2071
48.1 I2C introduction 2071
48.2 I2C main features 2071
48.3 I2C implementation 2072
48.4 I2C functional description 2072
48.4.1 I2C block diagram 2073
48.4.2 I2C pins and internal signals 2073
48.4.3 I2C clock requirements 2074
48.4.4 I2C mode selection 2074
48.4.5 I2C initialization 2075
48.4.6 I2C reset 2079

48.4.7 I2C data transfer . . . 2080
48.4.8 I2C target mode . . . 2082
48.4.9 I2C controller mode . . . 2091
48.4.10 I2C_TIMINGR register configuration examples . . . 2102
48.4.11 SMBus specific features . . . 2104
48.4.12 SMBus initialization . . . 2106
48.4.13 SMBus I2C_TIMEOUTR register configuration examples . . . 2108
48.4.14 SMBus target mode . . . 2109
48.4.15 SMBus controller mode . . . 2112
48.4.16 Wake-up from Stop mode on address match . . . 2115
48.4.17 Error conditions . . . 2116
48.5 I2C in low-power modes . . . 2118
48.6 I2C interrupts . . . 2118
48.7 I2C DMA requests . . . 2119
48.7.1 Transmission using DMA . . . 2119
48.7.2 Reception using DMA . . . 2119
48.8 I2C debug modes . . . 2119
48.9 I2C registers . . . 2120
48.9.1 I2C control register 1 (I2C_CR1) . . . 2120
48.9.2 I2C control register 2 (I2C_CR2) . . . 2123
48.9.3 I2C own address 1 register (I2C_OAR1) . . . 2125
48.9.4 I2C own address 2 register (I2C_OAR2) . . . 2125
48.9.5 I2C timing register (I2C_TIMINGR) . . . 2126
48.9.6 I2C timeout register (I2C_TIMEOUTR) . . . 2127
48.9.7 I2C interrupt and status register (I2C_ISR) . . . 2128
48.9.8 I2C interrupt clear register (I2C_ICR) . . . 2131
48.9.9 I2C PEC register (I2C_PECR) . . . 2132
48.9.10 I2C receive data register (I2C_RXDR) . . . 2132
48.9.11 I2C transmit data register (I2C_TXDR) . . . 2133
48.9.12 I2C register map . . . 2134

49 Improved inter-integrated circuit (I3C) . . . 2135
49.1 I3C introduction . . . 2135
49.2 I3C main features . . . 2135
49.3 I3C implementation . . . 2137
49.3.1 I3C instantiation . . . 2137

49.3.2 I3C wake-up from low-power mode(s) . . . 2137
49.3.3 I3C FIFOs . . . 2137
49.3.4 I3C triggers . . . 2137
49.3.5 I3C interrupt(s) . . . 2137
49.3.6 I3C MIPI® support . . . 2138
49.4 I3C block diagram . . . 2139
49.5 I3C pins and internal signals . . . 2139
49.6 I3C reset and clocks . . . 2140
49.6.1 I3C reset . . . 2140
49.6.2 I3C clocks and requirements . . . 2140
49.7 I3C peripheral state and programming . . . 2142
49.7.1 I3C peripheral state . . . 2142
49.7.2 I3C controller state and programming sequence . . . 2143
49.7.3 I3C target state and programming sequence . . . 2147
49.8 I3C registers and programming . . . 2151
49.8.1 I3C register set, as controller/target . . . 2151
49.8.2 I3C registers and fields use versus peripheral state, as controller . . . 2152
49.8.3 I3C registers and fields usage versus peripheral state, as target . . . 2155
49.9 I3C bus transfers and programming . . . 2157
49.9.1 I3C command set (CCCs), as controller/target . . . 2157
49.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller . . . 2161
49.9.3 I3C broadcast ENTDAA CCC transfer, as controller . . . 2163
49.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller . . . 2163
49.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target . . . 2165
49.9.6 I3C broadcast ENTDAA CCC transfer, as target . . . 2167
49.9.7 I3C broadcast DEFTGTS CCC transfer, as target . . . 2168
49.9.8 I3C broadcast DEFGRPA CCC transfer, as target . . . 2169
49.9.9 I3C direct GETSTATUS CCC response, as target . . . 2170
49.9.10 I3C private read/write transfer, as controller . . . 2171
49.9.11 I3C private read/write transfer, as target . . . 2172
49.9.12 Legacy I2C read/write transfer, as controller . . . 2173
49.9.13 I3C IBI transfer, as controller/target . . . 2174
49.9.14 I3C hot-join request transfer, as controller/target . . . 2175
49.9.15 I3C controller-role request transfer, as controller/target . . . 2176
49.10 I3C FIFOs management, as controller . . . 2177

49.10.1 C-FIFO management, as controller 2177
49.10.2 TX-FIFO management, as controller 2178
49.10.3 RX-FIFO management, as controller 2181
49.10.4 S-FIFO management, as controller 2183
49.11 I3C FIFOs management, as target 2185
49.11.1 RX-FIFO management, as target 2185
49.11.2 TX-FIFO management, as target 2186
49.12 I3C error management 2189
49.12.1 Controller error management 2189
49.12.2 Target error management 2191
49.13 I3C wake-up from low-power mode(s) 2192
49.13.1 Wake-up from Stop 2192
49.14 I3C in low-power modes 2195
49.15 I3C interrupts 2196
49.16 I3C registers 2197
49.16.1 I3C message control register (I3C_CR) 2197
49.16.2 I3C message control register [alternate] (I3C_CR) 2199
49.16.3 I3C configuration register (I3C_CFGR) 2201
49.16.4 I3C receive data byte register (I3C_RDR) 2206
49.16.5 I3C receive data word register (I3C_RDWR) 2206
49.16.6 I3C transmit data byte register (I3C_TDR) 2207
49.16.7 I3C transmit data word register (I3C_TDWR) 2208
49.16.8 I3C IBI payload data register (I3C_IBIDR) 2210
49.16.9 I3C target transmit configuration register (I3C_TGTTDR) 2211
49.16.10 I3C status register (I3C_SR) 2212
49.16.11 I3C status error register (I3C_SER) 2213
49.16.12 I3C received message register (I3C_RMR) 2215
49.16.13 I3C event register (I3C_EVR) 2216
49.16.14 I3C interrupt enable register (I3C_IER) 2220
49.16.15 I3C clear event register (I3C_CEVR) 2222
49.16.16 I3C own device characteristics register (I3C_DEVR0) 2224
49.16.17 I3C device x characteristics register (I3C_DEVRx) 2226
49.16.18 I3C maximum read length register (I3C_MAXRLR) 2228
49.16.19 I3C maximum write length register (I3C_MAXWLR) 2229
49.16.20 I3C timing register 0 (I3C_TIMINGR0) 2230
49.16.21 I3C timing register 1 (I3C_TIMINGR1) 2231

49.16.22 I3C timing register 2 (I3C_TIMINGR2) 2233
49.16.23 I3C bus characteristics register (I3C_BCR) 2234
49.16.24 I3C device characteristics register (I3C_DCR) 2235
49.16.25 I3C get capability register (I3C_GETCAPR) 2236
49.16.26 I3C controller-role capability register (I3C_CRCAPR) 2237
49.16.27 I3C get max data speed register (I3C_GETMXDSR) 2238
49.16.28 I3C extended provisioned ID register (I3C_EPIDR) 2240
49.16.29 I3C register map 2241

50 Universal synchronous/asynchronous receiver transmitter (USART/UART) 2244

50.1 USART introduction 2244
50.2 USART main features 2244
50.3 USART extended features 2245
50.4 USART implementation 2245
50.5 USART functional description 2247
50.5.1 USART block diagram 2247
50.5.2 USART pins and internal signals 2247
50.5.3 USART clocks 2249
50.5.4 USART character description 2249
50.5.5 USART FIFOs and thresholds 2252
50.5.6 USART transmitter 2252
50.5.7 USART receiver 2255
50.5.8 USART baud rate generation 2262
50.5.9 Tolerance of the USART receiver to clock deviation 2264
50.5.10 USART auto baud rate detection 2265
50.5.11 USART multiprocessor communication 2267
50.5.12 USART Modbus communication 2269
50.5.13 USART parity control 2270
50.5.14 USART LIN (local interconnection network) mode 2271
50.5.15 USART synchronous mode 2273
50.5.16 USART single-wire half-duplex communication 2277
50.5.17 USART receiver timeout 2277
50.5.18 USART smartcard mode 2278
50.5.19 USART IrDA SIR ENDEC block 2282
50.5.20 Continuous communication using USART and DMA 2285
50.5.21 RS232 hardware flow control and RS485 driver enable 2287

50.5.22 USART low-power management 2290
50.6 USART in low-power modes 2293
50.7 USART interrupts 2293
50.8 USART registers 2296
50.8.1 USART control register 1 (USART_CR1) 2296
50.8.2 USART control register 1 [alternate] (USART_CR1) 2300
50.8.3 USART control register 2 (USART_CR2) 2303
50.8.4 USART control register 3 (USART_CR3) 2307
50.8.5 USART control register 3 [alternate] (USART_CR3) 2311
50.8.6 USART baud rate register (USART_BRR) 2315
50.8.7 USART guard time and prescaler register (USART_GTPR) 2315
50.8.8 USART receiver timeout register (USART_RTOR) 2316
50.8.9 USART request register (USART_RQR) 2317
50.8.10 USART interrupt and status register (USART_ISR) 2318
50.8.11 USART interrupt and status register [alternate] (USART_ISR) 2324
50.8.12 USART interrupt flag clear register (USART_ICR) 2329
50.8.13 USART receive data register (USART_RDR) 2330
50.8.14 USART transmit data register (USART_TDR) 2331
50.8.15 USART prescaler register (USART_PRESC) 2331
50.8.16 USART register map 2332

51 Low-power universal asynchronous receiver transmitter (LPUART) 2334
51.1 LPUART introduction 2334
51.2 LPUART main features 2334
51.3 LPUART implementation 2335
51.4 LPUART functional description 2337
51.4.1 LPUART block diagram 2337
51.4.2 LPUART pins and internal signals 2338
51.4.3 LPUART clocks 2339
51.4.4 LPUART character description 2339
51.4.5 LPUART FIFOs and thresholds 2341
51.4.6 LPUART transmitter 2341
51.4.7 LPUART receiver 2345
51.4.8 LPUART baud rate generation 2349
51.4.9 Tolerance of the LPUART receiver to clock deviation 2350

51.4.10 LPUART multiprocessor communication 2351
51.4.11 LPUART parity control 2353
51.4.12 LPUART single-wire half-duplex communication 2354
51.4.13 Continuous communication using DMA and LPUART 2354
51.4.14 RS232 hardware flow control and RS485 driver enable 2357
51.4.15 LPUART low-power management 2359
51.5 LPUART in low-power modes 2362
51.6 LPUART interrupts 2363
51.7 LPUART registers 2364
51.7.1 LPUART control register 1 (LPUART_CR1) 2364
51.7.2 LPUART control register 1 [alternate] (LPUART_CR1) 2367
51.7.3 LPUART control register 2 (LPUART_CR2) 2370
51.7.4 LPUART control register 3 (LPUART_CR3) 2372
51.7.5 LPUART control register 3 [alternate] (LPUART_CR3) 2375
51.7.6 LPUART baud rate register (LPUART_BRR) 2377
51.7.7 LPUART request register (LPUART_RQR) 2377
51.7.8 LPUART interrupt and status register (LPUART_ISR) 2378
51.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 2383
51.7.10 LPUART interrupt flag clear register (LPUART_ICR) 2386
51.7.11 LPUART receive data register (LPUART_RDR) 2387
51.7.12 LPUART transmit data register (LPUART_TDR) 2387
51.7.13 LPUART prescaler register (LPUART_PRESC) 2388
51.7.14 LPUART register map 2389

52 Serial peripheral interface (SPI) 2391
52.1 SPI introduction 2391
52.2 SPI main features 2391
52.3 SPI implementation 2392
52.4 SPI functional description 2393
52.4.1 SPI block diagram 2393
52.4.2 SPI pins and internal signals 2394
52.4.3 SPI communication general aspects 2395
52.4.4 Communications between one master and one slave 2395
52.4.5 Standard multislave communication 2398
52.4.6 Multimaster communication 2401
52.4.7 Slave select (NSS pin) management 2402

52.4.8 Ready pin (RDY) management . . . 2406
52.4.9 Communication formats . . . 2406
52.4.10 Configuring the SPI . . . 2408
52.4.11 Enabling the SPI . . . 2409
52.4.12 SPI data transmission and reception procedures . . . 2410
52.4.13 Disabling the SPI . . . 2414
52.4.14 Communication using DMA (direct memory addressing) . . . 2415
52.5 SPI specific modes and control . . . 2417
52.5.1 TI mode . . . 2417
52.5.2 SPI error flags . . . 2417
52.5.3 CRC computation . . . 2421
52.6 SPI in low-power modes . . . 2422
52.7 SPI interrupts . . . 2422
52.8 I2S main features . . . 2424
52.9 I2S functional description . . . 2424
52.9.1 I2S general description . . . 2424
52.9.2 Pin sharing with SPI function . . . 2425
52.9.3 Bitfields usable in I2S/PCM mode . . . 2425
52.9.4 Slave and master modes . . . 2426
52.9.5 Supported audio protocols . . . 2426
52.9.6 Additional serial interface flexibility . . . 2432
52.9.7 Startup sequence . . . 2434
52.9.8 Stop sequence . . . 2436
52.9.9 Clock generator . . . 2437
52.9.10 Internal FIFOs . . . 2439
52.9.11 FIFO status flags . . . 2440
52.9.12 Handling of underrun situation . . . 2440
52.9.13 Handling of overrun situation . . . 2441
52.9.14 Frame error detection . . . 2442
52.9.15 DMA interface . . . 2444
52.9.16 Programing examples . . . 2444
52.10 I2S interrupts . . . 2447
52.11 SPI/I2S registers . . . 2447
52.11.1 SPI/I2S control register 1 (SPI_CR1) . . . 2447
52.11.2 SPI/I2S control register 2 (SPI_CR2) . . . 2449
52.11.3 SPI/I2S configuration register 1 (SPI_CFG1) . . . 2450

52.11.4 SPI/I2S configuration register 2 (SPI_CFG2) 2453
52.11.5 SPI/I2S interrupt enable register (SPI_IER) 2455
52.11.6 SPI/I2S status register (SPI_SR) 2456
52.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) 2459
52.11.8 SPI/I2S transmit data register (SPI_TXDR) 2460
52.11.9 SPI/I2S receive data register (SPI_RXDR) 2460
52.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) 2461
52.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) 2461
52.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) 2462
52.11.13 SPI/I2S underrun data register (SPI_UDRDR) 2463
52.11.14 SPI/I2S configuration register (SPI_I2SCFGR) 2463
52.11.15 SPI/I2S register map 2465

53 Serial audio interface (SAI) 2467

53.1 SAI introduction 2467
53.2 SAI main features 2467
53.3 SAI implementation 2468
53.4 SAI functional description 2469
53.4.1 SAI block diagram 2469
53.4.2 SAI pins and internal signals 2470
53.4.3 Main SAI modes 2471
53.4.4 SAI synchronization mode 2472
53.4.5 Audio data size 2473
53.4.6 Frame synchronization 2473
53.4.7 Slot configuration 2476
53.4.8 SAI clock generator 2478
53.4.9 Internal FIFOs 2481
53.4.10 PDM interface 2483
53.4.11 AC’97 link controller 2491
53.4.12 SPDIF output 2493
53.4.13 Specific features 2496
53.4.14 Error flags 2500
53.4.15 Disabling the SAI 2503
53.4.16 SAI DMA interface 2503
53.5 SAI interrupts 2504
53.6 SAI registers 2506

53.6.1 SAI global configuration register (SAI_GCR) 2506
53.6.2 SAI configuration register 1 (SAI_ACR1) 2506
53.6.3 SAI configuration register 2 (SAI_ACR2) 2509
53.6.4 SAI frame configuration register (SAI_AFRCR) 2511
53.6.5 SAI slot register (SAI_ASLOTR) 2512
53.6.6 SAI interrupt mask register (SAI_AIM) 2513
53.6.7 SAI status register (SAI_ASR) 2515
53.6.8 SAI clear flag register (SAI_ACLRFR) 2517
53.6.9 SAI data register (SAI_ADR) 2518
53.6.10 SAI configuration register 1 (SAI_BCR1) 2518
53.6.11 SAI configuration register 2 (SAI_BCR2) 2521
53.6.12 SAI frame configuration register (SAI_BFRCR) 2523
53.6.13 SAI slot register (SAI_BSLOTR) 2524
53.6.14 SAI interrupt mask register (SAI_BIM) 2525
53.6.15 SAI status register (SAI_BSR) 2526
53.6.16 SAI clear flag register (SAI_BCLRFR) 2528
53.6.17 SAI data register (SAI_BDR) 2529
53.6.18 SAI PDM control register (SAI_PDMCR) 2530
53.6.19 SAI PDM delay register (SAI_PDMDLY) 2531
53.6.20 SAI register map 2533

54 FD controller area network (FDCAN) 2535

54.1 FDCAN introduction 2535
54.2 FDCAN main features 2537
54.3 FDCAN functional description 2538
54.3.1 FDCAN block diagram 2538
54.3.2 FDCAN pins and internal signals 2539
54.3.3 Bit timing 2540
54.3.4 Operating modes 2541
54.3.5 Error management 2550
54.3.6 Message RAM 2551
54.3.7 FIFO acknowledge handling 2560
54.3.8 FDCAN Rx FIFO element 2560
54.3.9 FDCAN Tx buffer element 2562
54.3.10 FDCAN Tx event FIFO element 2564
54.3.11 FDCAN standard message ID filter element 2565
54.3.12 FDCAN extended message ID filter element 2566

54.4 FDCAN registers 2568
54.4.1 FDCAN core release register (FDCAN_CREL) 2568
54.4.2 FDCAN endian register (FDCAN_ENDN) 2568
54.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 2568
54.4.4 FDCAN test register (FDCAN_TEST) 2569
54.4.5 FDCAN RAM watchdog register (FDCAN_RWD) 2570
54.4.6 FDCAN CC control register (FDCAN_CCCR) 2571
54.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 2572
54.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 2574
54.4.9 FDCAN timestamp counter value register (FDCAN_TSCV) 2574
54.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 2575
54.4.11 FDCAN timeout counter value register (FDCAN_TOCV) 2576
54.4.12 FDCAN error counter register (FDCAN_ECR) 2576
54.4.13 FDCAN protocol status register (FDCAN_PSR) 2577
54.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 2579
54.4.15 FDCAN interrupt register (FDCAN_IR) 2579
54.4.16 FDCAN interrupt enable register (FDCAN_IE) 2582
54.4.17 FDCAN interrupt line select register (FDCAN_ILS) 2584
54.4.18 FDCAN interrupt line enable register (FDCAN_ILE) 2585
54.4.19 FDCAN global filter configuration register (FDCAN_RXGFC) 2585
54.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM) 2587
54.4.21 FDCAN high-priority message status register (FDCAN_HPMS) 2587
54.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 2588
54.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 2589
54.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 2589
54.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 2590
54.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) 2590
54.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 2591
54.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 2591
54.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) 2592
54.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 2593
54.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 2593
54.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 2594
54.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 2594
54.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 2595
54.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 2595

54.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 2596
54.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV) 2596
54.4.38 FDCAN register map 2597

55 Universal serial bus full-speed host/device interface (USB) 2601

55.1 USB introduction 2601
55.2 USB main features 2601
55.3 USB implementation 2601
55.4 USB functional description 2602
55.4.1 USB block diagram 2602
55.4.2 USB pins and internal signals 2602
55.4.3 USB reset and clocks 2603
55.4.4 General description and Device mode functionality 2603
55.4.5 Description of USB blocks used in both Device and Host modes . . . 2604
55.4.6 Description of host frame scheduler (HFS) specific to Host mode . . 2605
55.5 Programming considerations for Device and Host modes 2606
55.5.1 Generic USB Device programming 2606
55.5.2 System and power-on reset 2606
55.5.3 Double-buffered endpoints and usage in Device mode 2613
55.5.4 Double buffered channels: usage in Host mode 2615
55.5.5 Isochronous transfers in Device mode 2616
55.5.6 Isochronous transfers in Host mode 2617
55.5.7 Suspend/resume events 2618
55.6 USB registers 2622
55.6.1 USB control register (USB_CNTR) 2622
55.6.2 USB interrupt status register (USB_ISTR) 2625
55.6.3 USB frame number register (USB_FNR) 2629
55.6.4 USB Device address (USB_DADDR) 2629
55.6.5 USB LPM control and status register (USB_LPMCSR) 2630
55.6.6 USB battery charging detector (USB_BCDR) 2631
55.6.7 USB endpoint/channel n register (USB_CHEPnR) 2632
55.6.8 USB register map 2641
55.7 USBSRAM registers 2642
55.7.1 Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n) 2643
55.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n) 2643

55.7.3 Channel/endpoint receive buffer descriptor n (USB_CHEP_RXTXBD_n) . . . 2645
55.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USB_CHEP_RXTXBD_n) . . . 2646
55.7.5 USBSRAM register map . . . 2647

56 USB Type-C®/USB Power Delivery interface (UCPD) . . . 2648

56.1 UCPD introduction . . . 2648
56.2 UCPD main features . . . 2648
56.3 UCPD implementation . . . 2649
56.4 UCPD functional description . . . 2649
56.4.1 UCPD block diagram . . . 2650
56.4.2 UCPD reset and clocks . . . 2651
56.4.3 Physical layer protocol . . . 2652
56.4.4 UCPD BMC transmitter . . . 2658
56.4.5 UCPD BMC receiver . . . 2660
56.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd) . . . 2661
56.4.7 UCPD Type-C voltage monitoring and de-bouncing . . . 2662
56.4.8 UCPD fast role swap (FRS) . . . 2662
56.4.9 UCPD DMA Interface . . . 2662
56.4.10 Wake-up from Stop mode . . . 2662
56.5 UCPD programming sequences . . . 2663
56.5.1 Initialization phase . . . 2663
56.5.2 Type-C state machine handling . . . 2663
56.5.3 USB PD transmit . . . 2665
56.5.4 USB PD receive . . . 2666
56.5.5 UCPD software trimming . . . 2667
56.6 UCPD low-power modes . . . 2667
56.7 UCPD interrupts . . . 2668
56.8 UCPD registers . . . 2669
56.8.1 UCPD configuration register 1 (UCPD_CFGR1) . . . 2669
56.8.2 UCPD configuration register 2 (UCPD_CFGR2) . . . 2671
56.8.3 UCPD configuration register 3 (UCPD_CFGR3) . . . 2671
56.8.4 UCPD control register (UCPD_CR) . . . 2672
56.8.5 UCPD interrupt mask register (UCPD_IMR) . . . 2674
56.8.6 UCPD status register (UCPD_SR) . . . 2676
56.8.7 UCPD interrupt clear register (UCPD_ICR) . . . 2679

56.8.8 UCPD Tx ordered set type register (UCPD_TX_ORDSETR) 2680
56.8.9 UCPD Tx payload size register (UCPD_TX_PAYSZR) 2680
56.8.10 UCPD Tx data register (UCPD_TXDR) 2681
56.8.11 UCPD Rx ordered set register (UCPD_RX_ORDSETR) 2681
56.8.12 UCPD Rx payload size register (UCPD_RX_PAYSZR) 2682
56.8.13 UCPD receive data register (UCPD_RXDR) 2683
56.8.14 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1) 2683
56.8.15 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2) 2684
56.8.16 UCPD register map 2684

57 Ethernet (ETH): media access control (MAC) with DMA controller 2687

57.1 Ethernet introduction 2687
57.2 Ethernet main features 2687
57.2.1 Standard compliance 2687
57.2.2 MAC features 2687
57.2.3 Transaction layer (MTL) features 2689
57.2.4 DMA block features 2690
57.2.5 Bus interface features 2690
57.3 Ethernet pins and internal signals 2691
57.4 Ethernet architecture 2692
57.4.1 DMA controller 2693
57.4.2 MTL 2701
57.4.3 MAC 2702
57.5 Ethernet functional description: MAC 2707
57.5.1 Double VLAN processing 2707
57.5.2 Source address and VLAN insertion, replacement, or deletion 2708
57.5.3 Packet filtering 2710
57.5.4 IEEE 1588 timestamp support 2716
57.5.5 Checksum offload engine 2741
57.5.6 TCP segmentation offload 2747
57.5.7 IPv4 ARP offload 2753
57.5.8 Loopback 2754
57.5.9 Flow control 2755
57.5.10 MAC management counters 2758

57.5.11 Interrupts generated by the MAC . . . 2760
57.5.12 MAC and MMC register descriptions . . . 2760
57.6 Ethernet functional description: PHY interfaces . . . 2761
57.6.1 Station management agent (SMA) . . . 2761
57.6.2 Media independent interface (MII) . . . 2768
57.6.3 Reduced media independent interface (RMII) . . . 2769
57.7 Ethernet low-power modes . . . 2772
57.7.1 Low-power management . . . 2772
57.7.2 Energy Efficient Ethernet (EEE) . . . 2778
57.8 Ethernet interrupts . . . 2784
57.8.1 DMA interrupts . . . 2784
57.8.2 MTL interrupts . . . 2786
57.8.3 MAC Interrupts . . . 2786
57.9 Ethernet programming model . . . 2787
57.9.1 DMA initialization . . . 2787
57.9.2 MTL initialization . . . 2788
57.9.3 MAC initialization . . . 2788
57.9.4 Performing normal receive and transmit operation . . . 2789
57.9.5 Stopping and starting transmission . . . 2790
57.9.6 Programming guidelines for switching to new descriptor list in RxDMA . . . 2790
57.9.7 Programming guidelines for switching the AHB clock frequency . . . 2790
57.9.8 Programming guidelines for MII link state transitions . . . 2791
57.9.9 Programming guidelines for IEEE 1588 timestamping . . . 2792
57.9.10 Programming guidelines for PTP offload feature . . . 2793
57.9.11 Programming guidelines for Energy Efficient Ethernet (EEE) . . . 2797
57.9.12 Programming guidelines for flexible pulse-per-second (PPS) output 2799
57.9.13 Programming guidelines for IEEE 1588 auxiliary snapshot . . . 2801
57.9.14 Programming guidelines for TSO . . . 2801
57.9.15 Programming guidelines to perform VLAN filtering on the receiver . . 2802
57.10 Descriptors . . . 2803
57.10.1 Descriptor overview . . . 2803
57.10.2 Descriptor structure . . . 2803
57.10.3 Transmit descriptor . . . 2807
57.10.4 Receive descriptor . . . 2820
57.11 Ethernet registers . . . 2832
57.11.1 Ethernet register maps . . . 2832

57.11.2 Ethernet DMA registers 2832
57.11.3 Ethernet MTL registers 2856
57.11.4 Ethernet MAC and MMC registers 2868

58 HDMI-CEC controller (CEC) 2964
58.1 HDMI-CEC introduction 2964
58.2 HDMI-CEC controller main features 2964
58.3 HDMI-CEC functional description 2965
58.3.1 HDMI-CEC pin and internal signals 2965
58.3.2 HDMI-CEC block diagram 2966
58.3.3 Message description 2966
58.3.4 Bit timing 2967
58.4 Arbitration 2967
58.4.1 SFT option bit 2969
58.5 Error handling 2969
58.5.1 Bit error 2969
58.5.2 Message error 2970
58.5.3 Bit rising error (BRE) 2970
58.5.4 Short bit period error (SBPE) 2970
58.5.5 Long bit period error (LBPE) 2970
58.5.6 Transmission error detection (TXERR) 2972
58.6 HDMI-CEC interrupts 2973
58.7 HDMI-CEC registers 2974
58.7.1 CEC control register (CEC_CR) 2974
58.7.2 CEC configuration register (CEC_CFGR) 2975
58.7.3 CEC Tx data register (CEC_TXDR) 2977
58.7.4 CEC Rx data register (CEC_RXDR) 2977
58.7.5 CEC interrupt and status register (CEC_ISR) 2977
58.7.6 CEC interrupt enable register (CEC_IER) 2979
58.7.7 HDMI-CEC register map 2981

59 Debug support (DBG) 2982
59.1 DBG introduction 2982
59.2 DBG functional description 2983
59.2.1 DBG block diagram 2983
59.2.2 DBG pins and internal signals 2983

59.2.3 DBG reset and clocks . . . 2984
59.2.4 DBG power domains . . . 2984
59.2.5 Debug and low-power modes . . . 2984
59.2.6 Security . . . 2985
59.2.7 Debug authentication . . . 2987
59.3 Serial-wire and JTAG debug port (SWJ-DP) . . . 2990
59.3.1 JTAG debug port . . . 2991
59.3.2 Serial-wire debug port . . . 2993
59.3.3 Debug port registers . . . 2995
59.3.4 Debug port register map and reset values . . . 3001
59.4 Access ports . . . 3002
59.4.1 Access port registers . . . 3003
59.4.2 Access port register map . . . 3009
59.5 ROM tables . . . 3010
59.5.1 System ROM table registers . . . 3013
59.5.2 System ROM table register map . . . 3017
59.5.3 MCU ROM table registers . . . 3018
59.5.4 MCU ROM table register map . . . 3022
59.5.5 Processor ROM table registers . . . 3023
59.5.6 Processor ROM table register map . . . 3028
59.6 Data watchpoint and trace unit (DWT) . . . 3029
59.6.1 DWT registers . . . 3029
59.6.2 DWT register map . . . 3043
59.7 Instrumentation trace macrocell (ITM) . . . 3046
59.7.1 ITM registers . . . 3046
59.7.2 ITM register map . . . 3054
59.8 Breakpoint unit (BPU) . . . 3056
59.8.1 BPU registers . . . 3056
59.8.2 BPU register map . . . 3062
59.9 Embedded Trace Macrocell (ETM) . . . 3063
59.9.1 ETM registers . . . 3063
59.9.2 ETM register map . . . 3087
59.10 Trace port interface unit (TPIU) . . . 3091
59.10.1 TPIU registers . . . 3092
59.10.2 TPIU register map . . . 3101
59.11 Cross-trigger interface (CTI) . . . 3103

59.11.1 CTI registers 3104
59.11.2 CTI register map 3114
59.12 Microcontroller debug unit (DBGMCU) 3116
59.12.1 Device ID 3116
59.12.2 Low-power mode emulation 3116
59.12.3 Peripheral clock freeze 3117
59.12.4 DBGMCU registers 3118
59.12.5 DBGMCU register map 3132
59.13 References 3135

60 Device electronic signature 3136
60.1 Unique device ID register (96 bits) 3136
60.2 Flash size data register 3137
60.3 Package data register 3137

61 Important security notice 3139

62 Revision history 3140

List of tables

Table 1. Implementation of masters . . . 106
Table 2. Implementation of slaves . . . 106
Table 3. Securable peripherals by TZSC . . . 110
Table 4. TrustZone-aware peripherals . . . 114
Table 5. Memory map and peripheral register addresses (STM32H562/563/573xx) . . . 118
Table 6. Memory map and peripheral register addresses (STM32H523/533xx) . . . 123
Table 7. Configuring security attributes with IDAU and SAU . . . 136
Table 8. MPCWMx resources . . . 138
Table 9. MPCBBx resources (STM32H562/63/73xx devices) . . . 138
Table 10. MPCBBx resources (STM32H523/33xx devices) . . . 138
Table 11. DMA channel use (security) . . . 142
Table 12. Secure alternate function between peripherals and allocated I/Os . . . 144
Table 13. Nonsecure peripheral functions that cannot be connected to secure I/Os . . . 145
Table 14. Nonsecure peripheral functions that can be connected to secure I/Os . . . 145
Table 15. TrustZone-aware DBGMCU nonsecure accesses management . . . 146
Table 16. DMA channel use (privilege) . . . 150
Table 17. Internal tampers in TAMP . . . 154
Table 18. Effect of low-power modes on TAMP . . . 155
Table 19. Accelerated cryptographic operations . . . 158
Table 20. Main product life cycle transitions . . . 160
Table 21. Typical product life cycle phases . . . 161
Table 22. Software intellectual property protection with PRODUCT_STAT . . . 166
Table 23. Boot mode when TrustZone is disabled (TZEN = 0xC3) - STM32H523/62/63xx devices 170
Table 24. Boot mode when TrustZone is enabled (TZEN = 0xB4) - STM32H523/62/63xx devices . 170
Table 25. Boot mode when TrustZone is disabled (TZEN = 0xC3) - STM32H533/73xx . . . 171
Table 26. Boot mode when TrustZone is enabled (TZEN = 0xB4) - STM32H533/73xx . . . 171
Table 27. GTZC features . . . 174
Table 28. GTZC1 sub-block address offset . . . 175
Table 29. MPCWM resource assignment . . . 175
Table 30. MPCWM3 and MPCWM4 (subregions A and B) . . . 175
Table 31. MPCBB resource assignment (STM32H562/63/73xx devices) . . . 175
Table 32. MPCBB resource assignment (STM32H523/33xx devices) . . . 176
Table 33. Secure properties of subregions A and B . . . 179
Table 34. Privileged properties of subregions A and B . . . 179
Table 35. GTZC interrupt request . . . 180
Table 36. GTZC1 TZSC register map and reset values . . . 199
Table 37. GTZC1 TZIC register map and reset values . . . 227
Table 38. GTZC1 MPCBBz register map and reset values (z = 1 to 3) . . . 231
Table 39. SRAMs density (Kbytes) . . . 232
Table 40. Internal SRAMs features . . . 233
Table 41. SRAM2 features . . . 235
Table 42. Effect of low-power modes on RAMCFG . . . 236
Table 43. RAMCFG interrupt requests . . . 236
Table 44. RAMCFG register map and reset values . . . 243
Table 45. Recommended number of wait states and programming delay . . . 252
Table 46. Flash memory OTP organization . . . 261
Table 47. Read-only public data organization . . . 262
Table 48. Memory map and swapping options (STM32H562/563/573xx devices) . . . 265

Table 49. Memory map and swapping options (STM32H523/533xx devices) . . . 265
Table 50. Recommended reactions to FLASH_OPSR contents . . . 268
Table 51. Option bytes organization . . . 271
Table 52. Specific modifying rules . . . 275
Table 53. OB modifiable in closed product . . . 275
Table 54. Option bytes key area . . . 277
Table 55. Default secure watermark . . . 283
Table 56. Flash memory TZ protection summary . . . 283
Table 57. TZ protection and bank or mass erase summary . . . 284
Table 58. Secure watermark-based area . . . 285
Table 59. Secure hide protection . . . 286
Table 60. HDP protections summary . . . 287
Table 61. Secure configuration block-based registers access conditions . . . 287
Table 62. Privilege protection summary . . . 288
Table 63. Privilege and mass or bank erase . . . 288
Table 64. Privilege configuration register access conditions (TZ enabled) . . . 288
Table 65. Privilege configuration register access conditions (TZ disabled) . . . 289
Table 66. Flash register accesses . . . 289
Table 67. Flash interface register protection summary . . . 292
Table 68. High-cycle area protection summary: access to data area address range . . . 293
Table 69. HDP protected definition . . . 293
Table 70. Privileged sectors and data area - Access to data area address range . . . 293
Table 71. Product states, debug states and debug policy . . . 294
Table 72. PRODUCT_STATE transitions . . . 296
Table 73. TZ OBK protection summary . . . 297
Table 74. OBK protection summary with TZ disabled . . . 298
Table 75. Access conditions to secure control register . . . 298
Table 76. Access conditions to non-secure control register . . . 298
Table 77. OTP/RO access constraints . . . 299
Table 78. RSSLIB/NSSLIB accesses . . . 299
Table 79. RSSLIB/NSSLIB entry point access . . . 300
Table 80. RSS lib interface functions . . . 300
Table 81. NSS lib interface functions . . . 304
Table 82. Effect of low-power modes on the embedded flash memory . . . 306
Table 83. Locating ECC failure . . . 313
Table 84. Flash interrupt request . . . 316
Table 85. Register map and reset value table . . . 361
Table 86. ICACHE features . . . 367
Table 87. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 369
Table 88. TAG memory dimensioning parameters for direct-mapped cache mode . . . 370
Table 89. ICACHE cacheability for AHB transaction . . . 372
Table 90. Memory configurations . . . 372
Table 91. ICACHE remap region size, base address, and remap address . . . 373
Table 92. ICACHE interrupts . . . 377
Table 93. ICACHE register map and reset values . . . 382
Table 94. DCACHE features . . . 384
Table 95. TAG memory dimensioning parameters . . . 387
Table 96. DCACHE cacheability for AHB transaction . . . 389
Table 97. DCACHE interrupts . . . 394
Table 98. DCACHE register map and reset values . . . 400
Table 99. PWR input/output pins . . . 403

Table 100. PWR internal input/output signals . . . 403
Table 101. Low-power mode summary . . . 420
Table 102. Functionalities depending on the working mode . . . 420
Table 103. Sleep mode . . . 424
Table 104. Memory shut-off block selection . . . 425
Table 105. Stop mode . . . 427
Table 106. Standby mode . . . 430
Table 107. Power modes output states versus MCU power modes . . . 430
Table 108. PWR security configuration summary . . . 431
Table 109. PWR interrupt requests . . . 433
Table 110. PWR register map and reset values . . . 450
Table 111. RCC input/output signals connected to package pins or balls . . . 452
Table 112. Reset source identification (RCC_RSR) . . . 454
Table 113. STOPWUCK and STOPKERWUCK description . . . 466
Table 114. HSIKERON and CSIKERON behavior . . . 467
Table 115. Kernel clock distribution overview . . . 469
Table 116. RCC security configuration summary . . . 475
Table 117. Interrupt sources and control . . . 479
Table 118. RCC register map and reset values . . . 557
Table 119. CRS features . . . 563
Table 120. CRS internal input/output signals . . . 564
Table 121. CRS interconnection . . . 565
Table 122. Effect of low-power modes on CRS . . . 568
Table 123. Interrupt control bits . . . 568
Table 124. CRS register map and reset values . . . 573
Table 125. Port bit configuration . . . 576
Table 126. GPIO secured bits . . . 584
Table 127. GPIO register map and reset values . . . 594
Table 128. SBS internal input/output signals . . . 598
Table 129. HDPL encoded values . . . 602
Table 130. SBS boot logic . . . 602
Table 131. OBK-HDPL logic . . . 607
Table 132. SBS register map and reset values . . . 620
Table 133. Peripherals interconnect matrix . . . 622
Table 134. GPDMA1/2 channel implementation . . . 633
Table 135. GPDMA1/2 wake-up in low-power modes . . . 633
Table 136. Programmed GPDMA1/2 request . . . 633
Table 137. Programmed GPDMA1/2 request as a block request . . . 638
Table 138. GPDMA1/2 channel with peripheral early termination . . . 638
Table 139. Programmed GPDMA1/2 request with peripheral early termination . . . 638
Table 140. Programmed GPDMA1/2 trigger . . . 639
Table 141. Programmed GPDMA source/destination burst . . . 660
Table 142. Programmed data handling . . . 665
Table 143. Effect of low-power modes on GPDMA . . . 678
Table 144. GPDMA interrupt requests . . . 679
Table 145. GPDMA register map and reset values . . . 708
Table 146. STM32H562/563/573xx vector table . . . 712
Table 147. STM32H523/533xx vector table . . . 717
Table 148. EXTI signals . . . 723
Table 149. EVG signals . . . 724
Table 150. EXTI line connections . . . 724
Table 151. Masking functionality . . . 728

Table 152. Register protection overview . . . 729
Table 153. EXTI register map sections . . . 731
Table 154. EXTI register map and reset values . . . 754
Table 155. CRC internal input/output signals . . . 757
Table 156. CRC register map and reset values . . . 762
Table 157. CORDIC functions . . . 764
Table 158. Cosine parameters . . . 765
Table 159. Sine parameters . . . 765
Table 160. Phase parameters . . . 766
Table 161. Modulus parameters . . . 766
Table 162. Arctangent parameters . . . 767
Table 163. Hyperbolic cosine parameters . . . 767
Table 164. Hyperbolic sine parameters . . . 768
Table 165. Hyperbolic arctangent parameters . . . 768
Table 166. Natural logarithm parameters . . . 769
Table 167. Natural log scaling factors and corresponding ranges . . . 769
Table 168. Square root parameters . . . 770
Table 169. Square root scaling factors and corresponding ranges . . . 770
Table 170. Precision vs. number of iterations . . . 773
Table 171. CORDIC register map and reset value . . . 780
Table 172. Valid combinations for read and write methods . . . 794
Table 173. FMAC register map and reset values . . . 807
Table 174. NOR/PSRAM bank selection . . . 814
Table 175. NOR/PSRAM External memory address . . . 814
Table 176. NAND memory mapping and timing registers . . . 814
Table 177. NAND bank selection . . . 815
Table 178. SDRAM bank selection . . . 815
Table 179. SDRAM address mapping . . . 815
Table 180. SDRAM address mapping with 8-bit data bus width . . . 816
Table 181. SDRAM address mapping with 16-bit data bus width . . . 817
Table 182. Programmable NOR/PSRAM access parameters . . . 818
Table 183. Non-multiplexed I/O NOR flash memory . . . 819
Table 184. 16-bit multiplexed I/O NOR flash memory . . . 819
Table 185. Non-multiplexed I/Os PSRAM/SRAM . . . 820
Table 186. 16-Bit multiplexed I/O PSRAM . . . 820
Table 187. NOR flash/PSRAM: example of supported memories and transactions . . . 821
Table 188. FMC_BCRx bitfields (mode 1) . . . 824
Table 189. FMC_BTRx bitfields (mode 1) . . . 824
Table 190. FMC_BCRx bitfields (mode A) . . . 826
Table 191. FMC_BTRx bitfields (mode A) . . . 826
Table 192. FMC_BWTRx bitfields (mode A) . . . 827
Table 193. FMC_BCRx bitfields (mode 2/B) . . . 829
Table 194. FMC_BTRx bitfields (mode 2/B) . . . 829
Table 195. FMC_BWTRx bitfields (mode 2/B) . . . 830
Table 196. FMC_BCRx bitfields (mode C) . . . 831
Table 197. FMC_BTRx bitfields (mode C) . . . 832
Table 198. FMC_BWTRx bitfields (mode C) . . . 832
Table 199. FMC_BCRx bitfields (mode D) . . . 834
Table 200. FMC_BTRx bitfields (mode D) . . . 835
Table 201. FMC_BWTRx bitfields (mode D) . . . 835
Table 202. FMC_BCRx bitfields (Muxed mode) . . . 837

Table 203. FMC_BTRx bitfields (Muxed mode) . . . 838
Table 204. FMC_BCRx bitfields (Synchronous multiplexed read mode) . . . 843
Table 205. FMC_BTRx bitfields (Synchronous multiplexed read mode) . . . 844
Table 206. FMC_BCRx bitfields (Synchronous multiplexed write mode) . . . 845
Table 207. FMC_BTRx bitfields (Synchronous multiplexed write mode) . . . 846
Table 208. Programmable NAND flash access parameters . . . 855
Table 209. 8-bit NAND flash . . . 855
Table 210. 16-bit NAND flash . . . 856
Table 211. Supported memories and transactions . . . 856
Table 212. ECC result relevant bits . . . 865
Table 213. SDRAM signals . . . 866
Table 214. FMC register map and reset values . . . 882
Table 215. OCTOSPI implementation . . . 886
Table 216. OCTOSPI input/output pins . . . 888
Table 217. OCTOSPI internal signals . . . 888
Table 218. Command/address phase description . . . 897
Table 219. OctaRAM command address bit assignment (based on 64-Mbyte OctaRAM) . . . 907
Table 220. Address alignment cases . . . 914
Table 221. OCTOSPI interrupt requests . . . 915
Table 222. OCTOSPI register map and reset values . . . 938
Table 223. SDMMC instances on device . . . 941
Table 224. SDMMC features . . . 942
Table 225. SDMMC operation modes SD and SDIO . . . 944
Table 226. SDMMC operation modes e•MMC . . . 944
Table 227. SDMMC internal input/output signals . . . 945
Table 228. SDMMC pins . . . 946
Table 229. SDMMC Command and data phase selection . . . 947
Table 230. Command token format . . . 953
Table 231. Short response with CRC token format . . . 954
Table 232. Short response without CRC token format . . . 954
Table 233. Long response with CRC token format . . . 954
Table 234. Specific Commands overview . . . 955
Table 235. Command path status flags . . . 956
Table 236. Command path error handling . . . 956
Table 237. Data token format . . . 964
Table 238. Data path status flags and clear bits . . . 964
Table 239. Data path error handling . . . 966
Table 240. Data FIFO access . . . 967
Table 241. Transmit FIFO status flags . . . 968
Table 242. Receive FIFO status flags . . . 969
Table 243. AHB and SDMMC_CK clock frequency relation . . . 974
Table 244. SDIO special operation control . . . 974
Table 245. 4-bit mode Start, interrupt, and CRC-status Signaling detection . . . 978
Table 246. CMD12 use cases . . . 983
Table 247. SDMMC interrupts . . . 997
Table 248. Response type and SDMMC_RESPxR registers . . . 1004
Table 249. SDMMC register map . . . 1020
Table 250. STM32H5 features . . . 1023
Table 251. DLYB internal input/output signals . . . 1024
Table 252. Delay block control . . . 1025
Table 253. DLYB register map and reset values . . . 1028

Table 254. ADC features . . . 1031
Table 255. ADC input/output pins . . . 1033
Table 256. ADC internal input/output signals . . . 1033
Table 257. ADC interconnection . . . 1033
Table 258. Configuring the trigger polarity for regular external triggers . . . 1053
Table 259. Configuring the trigger polarity for injected external triggers . . . 1054
Table 260. TSAR timings depending on resolution . . . 1066
Table 261. Offset computation versus data resolution . . . 1069
Table 262. Analog watchdog channel selection . . . 1080
Table 263. Analog watchdog 1 comparison . . . 1081
Table 264. Analog watchdog 2 and 3 comparison . . . 1081
Table 265. Maximum output results versus N and M (gray cells indicate truncation) . . . 1085
Table 266. Oversampler operating modes summary . . . 1090
Table 267. ADC interrupts . . . 1109
Table 268. DELAY bits versus ADC resolution . . . 1144
Table 269. ADC global register map . . . 1146
Table 270. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) . . . 1146
Table 271. ADC register map and reset values (master and slave ADC common registers) . . . 1148
Table 272. DTS internal input/output signals . . . 1151
Table 273. Sampling time configuration . . . 1154
Table 274. Trigger configuration . . . 1155
Table 275. Temperature sensor behavior in low-power modes . . . 1157
Table 276. Interrupt control bits . . . 1158
Table 277. DTS register map and reset values . . . 1166
Table 278. DAC features . . . 1168
Table 279. DAC input/output pins . . . 1170
Table 280. DAC internal input/output signals . . . 1170
Table 281. DAC interconnection . . . 1170
Table 282. Data format (case of 12-bit data) . . . 1173
Table 283. HFSEL description . . . 1173
Table 284. Sample and refresh timings . . . 1179
Table 285. Channel output modes summary . . . 1181
Table 286. Effect of low-power modes on DAC . . . 1187
Table 287. DAC interrupts . . . 1188
Table 288. DAC register map and reset values . . . 1204
Table 289. VREFBUF typical values . . . 1207
Table 290. VREF buffer modes . . . 1208
Table 291. VREFBUF register map and reset values . . . 1210
Table 292. DCMI input/output pins . . . 1212
Table 293. DCMI internal input/output signals . . . 1212
Table 294. Positioning of captured data bytes in 32-bit words (8-bit width) . . . 1214
Table 295. Positioning of captured data bytes in 32-bit words (10-bit width) . . . 1214
Table 296. Positioning of captured data bytes in 32-bit words (12-bit width) . . . 1214
Table 297. Positioning of captured data bytes in 32-bit words (14-bit width) . . . 1215
Table 298. Data storage in monochrome progressive video format . . . 1220
Table 299. Data storage in RGB progressive video format . . . 1221
Table 300. Data storage in YCbCr progressive video format . . . 1221
Table 301. Data storage in YCbCr progressive video format - Y extraction mode . . . 1221
Table 302. DCMI interrupts . . . 1222
Table 303. DCMI register map and reset values . . . 1232

Table 304. PSSI input/output pins . . . 1235
Table 305. PSSI internal input/output signals . . . 1235
Table 306. Positioning of captured data bytes in 32-bit words (8-bit width) . . . 1236
Table 307. Positioning of captured data bytes in 32-bit words (16-bit width) . . . 1237
Table 308. PSSI interrupt requests . . . 1240
Table 309. PSSI register map and reset values . . . 1246
Table 310. RNG internal input/output signals . . . 1248
Table 311. RNG interrupt requests . . . 1256
Table 312. RNG initialization times . . . 1257
Table 313. RNG configurations . . . 1257
Table 314. Configuration selection . . . 1258
Table 315. RNG register map and reset map . . . 1263
Table 316. AES versus SAES features . . . 1265
Table 317. AES internal input/output signals . . . 1266
Table 318. AES approved symmetric key functions . . . 1267
Table 319. Counter mode initialization vector definition . . . 1276
Table 320. Initialization of IV registers in GCM mode . . . 1279
Table 321. GCM last block definition . . . 1279
Table 322. Initialization of IV registers in CCM mode . . . 1286
Table 323. AES data swapping example . . . 1289
Table 324. Key endianness in AES_KEYRx registers (128/256-bit keys) . . . 1291
Table 325. IVI bitfield spread over AES_IVRx registers . . . 1292
Table 326. AES interrupt requests . . . 1293
Table 327. Processing latency for ECB, CBC and CTR . . . 1294
Table 328. Processing latency for GCM and CCM (in clock cycles) . . . 1295
Table 329. AES register map and reset values . . . 1307
Table 330. AES versus SAES features . . . 1311
Table 331. SAES internal input/output signals . . . 1312
Table 332. SAES approved symmetric key functions . . . 1313
Table 333. Counter mode initialization vector definition . . . 1323
Table 334. Initialization of IV registers in GCM mode . . . 1326
Table 335. GCM last block definition . . . 1326
Table 336. Initialization of IV registers in CCM mode . . . 1332
Table 337. AES data swapping example . . . 1341
Table 338. Key endianness in SAES_KEYRx registers (128/256-bit keys) . . . 1343
Table 339. IVI bitfield spread over SAES_IVRx registers . . . 1345
Table 340. SAES interrupt requests . . . 1347
Table 341. Processing latency for ECB, CBC and CTR . . . 1348
Table 342. Processing latency for GCM and CCM (in SAES kernel clock cycles) . . . 1348
Table 343. SAES register map and reset values . . . 1363
Table 344. HASH internal input/output signals . . . 1366
Table 345. Information on supported hash algorithms . . . 1367
Table 346. Hash processor outputs . . . 1370
Table 347. Processing time (in clock cycle) . . . 1376
Table 348. HASH interrupt requests . . . 1377
Table 349. HASH1 register map and reset values . . . 1385
Table 350. Internal input/output signals . . . 1388
Table 351. PKA integer arithmetic functions list . . . 1389
Table 352. PKA prime field (Fp) elliptic curve functions list . . . 1390
Table 353. Montgomery parameter computation . . . 1396
Table 354. Modular addition . . . 1397
Table 355. Modular subtraction . . . 1397

Table 356. Montgomery multiplication . . . 1398
Table 357. Modular exponentiation (normal mode) . . . 1399
Table 358. Modular exponentiation (fast mode) . . . 1399
Table 359. Modular exponentiation (protected mode) . . . 1400
Table 360. Modular inversion . . . 1400
Table 361. Modular reduction . . . 1401
Table 362. Arithmetic addition . . . 1401
Table 363. Arithmetic subtraction . . . 1401
Table 364. Arithmetic multiplication . . . 1402
Table 365. Arithmetic comparison . . . 1402
Table 366. CRT exponentiation . . . 1403
Table 367. Point on elliptic curve Fp check . . . 1404
Table 368. ECC Fp scalar multiplication . . . 1404
Table 369. ECDSA sign - Inputs . . . 1406
Table 370. ECDSA sign - Outputs . . . 1406
Table 371. Extended ECDSA sign - Extra outputs . . . 1407
Table 372. ECDSA verification - Inputs . . . 1407
Table 373. ECDSA verification - Outputs . . . 1408
Table 374. ECC complete addition . . . 1408
Table 375. ECC double base ladder . . . 1409
Table 376. ECC projective to affine . . . 1410
Table 377. Family of supported curves for ECC operations . . . 1411
Table 378. Modular exponentiation . . . 1412
Table 379. ECC scalar multiplication . . . 1412
Table 380. ECDSA signature average computation time . . . 1413
Table 381. ECDSA verification average computation times . . . 1413
Table 382. ECC double base ladder average computation times . . . 1413
Table 383. ECC projective to affine average computation times . . . 1413
Table 384. ECC complete addition average computation times . . . 1413
Table 385. Point on elliptic curve Fp check average computation times . . . 1413
Table 386. Montgomery parameters average computation times . . . 1414
Table 387. PKA interrupt requests . . . 1414
Table 388. PKA register map and reset values . . . 1419
Table 389. OTFDEC internal input/output signals . . . 1421
Table 390. OTFDEC interrupt requests . . . 1425
Table 391. OTFDEC register map and reset values . . . 1439
Table 392. TIM input/output pins . . . 1445
Table 393. TIM internal input/output signals . . . 1445
Table 394. Interconnect to the tim_ti1 input multiplexer . . . 1447
Table 395. Interconnect to the tim_ti2 input multiplexer . . . 1447
Table 396. Interconnect to the tim_ti3 input multiplexer . . . 1447
Table 397. Interconnect to the tim_ti4 input multiplexer . . . 1447
Table 398. Internal trigger connection . . . 1447
Table 399. Interconnect to the tim_etr input multiplexer . . . 1448
Table 400. Timer break interconnect . . . 1448
Table 401. Timer break2 interconnect . . . 1448
Table 402. System break interconnect . . . 1449
Table 403. CCR and ARR register change dithering pattern . . . 1482
Table 404. CCR register change dithering pattern in center-aligned PWM mode . . . 1483
Table 405. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 1495
Table 406. Break protection disarming conditions . . . 1497
Table 407. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 1506

Table 408. Counting direction versus encoder signals and polarity settings . . . 1510
Table 409. DMA request . . . 1531
Table 410. Effect of low-power modes on TIM1/TIM8 . . . 1532
Table 411. Interrupt requests . . . 1532
Table 412. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 1559
Table 413. TIMx register map and reset values . . . 1582
Table 414. STM32H5 general purpose timers . . . 1586
Table 415. TIM input/output pins . . . 1588
Table 416. TIM internal input/output signals . . . 1588
Table 417. Interconnect to the tim_ti1 input multiplexer . . . 1589
Table 418. Interconnect to the tim_ti2 input multiplexer . . . 1589
Table 419. Interconnect to the tim_ti3 input multiplexer . . . 1589
Table 420. Interconnect to the tim_ti4 input multiplexer . . . 1590
Table 421. TIMx internal trigger connection . . . 1590
Table 422. Interconnect to the tim_etr input multiplexer . . . 1590
Table 423. CCR and ARR register change dithering pattern . . . 1622
Table 424. CCR register change dithering pattern in center-aligned PWM mode . . . 1623
Table 425. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 1632
Table 426. Counting direction versus encoder signals and polarity settings . . . 1637
Table 427. DMA request . . . 1661
Table 428. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5 . . . 1661
Table 429. Interrupt requests . . . 1662
Table 430. Output control bit for standard tim_ocx channels . . . 1683
Table 431. TIM2/TIM3/TIM4/TIM5 register map and reset values . . . 1699
Table 432. TIM internal input/output signals . . . 1703
Table 433. TIMx_ARR register change dithering pattern . . . 1713
Table 434. DMA request . . . 1714
Table 435. Effect of low-power modes on TIM6/TIM7 . . . 1714
Table 436. Interrupt request . . . 1714
Table 437. TIMx register map and reset values . . . 1720
Table 438. TIM input/output pins . . . 1724
Table 439. TIM internal input/output signals . . . 1724
Table 440. Interconnect to the tim_ti1 input multiplexer . . . 1726
Table 441. Interconnect to the tim_ti2 input multiplexer . . . 1726
Table 442. TIMx internal trigger connection . . . 1726
Table 443. CCR and ARR register change dithering pattern . . . 1744
Table 444. Effect of low-power modes on TIM12/TIM13/TIM14 . . . 1753
Table 445. Interrupt requests . . . 1753
Table 446. Output control bit for standard tim_ocx channels . . . 1766
Table 447. TIM12 register map and reset values . . . 1770
Table 448. Output control bit for standard tim_ocx channels . . . 1779
Table 449. TIM13/TIM14 register map and reset values . . . 1782
Table 450. TIM input/output pins . . . 1787
Table 451. TIM internal input/output signals . . . 1788
Table 452. Interconnect to the tim_ti1 input multiplexer . . . 1789
Table 453. Interconnect to the tim_ti2 input multiplexer . . . 1789
Table 454. TIMx internal trigger connection . . . 1789
Table 455. Timer break interconnect . . . 1790
Table 456. System break interconnect . . . 1790
Table 457. CCR and ARR register change dithering pattern . . . 1810
Table 458. Break protection disarming conditions . . . 1819

Table 459. DMA request . . . 1830
Table 460. Effect of low-power modes on TIM15/TIM16/TIM17 . . . 1831
Table 461. Interrupt requests . . . 1831
Table 462. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) . . . 1847
Table 463. TIM15 register map and reset values . . . 1860
Table 464. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 1873
Table 465. TIM16/TIM17 register map and reset values . . . 1887
Table 466. STM32H5 LPTIM features . . . 1891
Table 467. LPTIM1/2/3/5/6 input/output pins . . . 1893
Table 468. LPTIM4 input/output pins . . . 1893
Table 469. LPTIM1/2/3/5/6 internal signals . . . 1894
Table 470. LPTIM4 internal signals . . . 1894
Table 471. LPTIM1/2/3/4/5/6 external trigger connections . . . 1895
Table 472. LPTIM1/2/3/5/6 input 1 connections . . . 1895
Table 473. LPTIM1/2/3/5/6 input 2 connections . . . 1895
Table 474. LPTIM1/2/3/5/6 input capture 1 connections . . . 1895
Table 475. LPTIM1 input capture 2 connections . . . 1896
Table 476. LPTIM2 input capture 2 connections . . . 1896
Table 477. LPTIM3/5/6 input capture 2 connections . . . 1896
Table 478. Prescaler division ratios . . . 1898
Table 479. Encoder counting scenarios . . . 1905
Table 480. Input capture Glitch filter latency (in counter step unit) . . . 1909
Table 481. Effect of low-power modes on the LPTIM . . . 1913
Table 482. Interrupt events . . . 1914
Table 483. LPTIM register map and reset values . . . 1938
Table 484. IWDG features . . . 1941
Table 485. IWDG delays versus actions . . . 1942
Table 486. IWDG internal input/output signals . . . 1943
Table 487. Effect of low power modes on IWDG . . . 1948
Table 488. IWDG interrupt request . . . 1950
Table 489. IWDG register map and reset values . . . 1956
Table 490. WWDG features . . . 1957
Table 491. WWDG internal input/output signals . . . 1958
Table 492. WWDG interrupt requests . . . 1961
Table 493. WWDG register map and reset values . . . 1963
Table 494. RTC input/output pins . . . 1967
Table 495. RTC internal input/output signals . . . 1967
Table 496. RTC interconnection . . . 1968
Table 497. RTC pin PC13 configuration . . . 1969
Table 498. PI8 configuration . . . 1971
Table 499. RTC_OUT mapping . . . 1972
Table 500. Effect of low-power modes on RTC . . . 1987
Table 501. RTC pins functionality over modes . . . 1987
Table 502. Nonsecure interrupt requests . . . 1988
Table 503. Secure interrupt requests . . . 1988
Table 504. RTC register map and reset values . . . 2019
Table 505. TAMP input/output pins . . . 2025
Table 506. TAMP internal input/output signals . . . 2025
Table 507. TAMP interconnection . . . 2026
Table 508. Device resource x tamper protection . . . 2032

Table 509. Active tamper output change period . . . 2035
Table 510. Minimum ATPER value . . . 2036
Table 511. Active tamper filtered pulse duration . . . 2037
Table 512. Effect of low-power modes on TAMP . . . 2038
Table 513. TAMP pins functionality over modes . . . 2039
Table 514. Interrupt requests . . . 2039
Table 515. TAMP register map and reset values . . . 2069
Table 516. I2C implementation . . . 2072
Table 517. I2C input/output pins . . . 2073
Table 518. I2C internal input/output signals . . . 2074
Table 519. Comparison of analog and digital filters . . . 2076
Table 520. I²C-bus and SMBus specification data setup and hold times . . . 2078
Table 521. I2C configuration . . . 2082
Table 522. I²C-bus and SMBus specification clock timings . . . 2093
Table 523. Timing settings for f_{I2CCLK} of 8 MHz . . . 2103
Table 524. Timing settings for f_{I2CCLK} of 16 MHz . . . 2103
Table 525. SMBus timeout specifications . . . 2105
Table 526. SMBus with PEC configuration . . . 2107
Table 527. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 2108
Table 528. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 2108
Table 529. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 2108
Table 530. Effect of low-power modes to I2C . . . 2118
Table 531. I2C interrupt requests . . . 2118
Table 532. I2C register map and reset values . . . 2134
Table 533. I3C wake-up . . . 2137
Table 534. I3C FIFOs implementation . . . 2137
Table 535. I3C interrupt(s) . . . 2137
Table 536. I3C peripheral controller/target features versus MIPI v1.1 . . . 2138
Table 537. I3C input/output pins . . . 2139
Table 538. I3C internal input/output signals . . . 2139
Table 539. I3C register usage . . . 2151
Table 540. I3C registers/fields usage versus controller state . . . 2152
Table 541. I3C registers/fields usage versus target state . . . 2155
Table 542. List of supported I3C CCCs, as controller/target . . . 2158
Table 543. I3C controller error management . . . 2189
Table 544. I3C target error management . . . 2191
Table 545. Effect of low-power modes . . . 2195
Table 546. I3C interrupt requests . . . 2196
Table 547. I3C register map and reset values . . . 2241
Table 548. Instance implementation . . . 2245
Table 549. USART/LPUART features . . . 2246
Table 550. USART/UART input/output pins . . . 2248
Table 551. USART internal input/output signals . . . 2249
Table 552. Noise detection from sampled data . . . 2261
Table 553. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 2265
Table 554. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 2265
Table 555. USART frame formats . . . 2270
Table 556. Effect of low-power modes on the USART . . . 2293
Table 557. USART interrupt requests . . . 2294
Table 558. USART register map and reset values . . . 2332
Table 559. Instance implementation on STM32H5 . . . 2335
Table 560. USART/LPUART features . . . 2335

Table 561. LPUART input/output pins . . . 2338
Table 562. LPUART internal input/output signals . . . 2338
Table 563. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 2349
Table 564. Tolerance of the LPUART receiver . . . 2350
Table 566. Effect of low-power modes on the LPUART . . . 2362
Table 567. LPUART interrupt requests . . . 2363
Table 568. LPUART register map and reset values . . . 2389
Table 569. SPI features . . . 2392
Table 570. SPI/I2S input/output pins . . . 2395
Table 571. SPI internal input/output signals . . . 2395
Table 572. Effect of low-power modes on the SPI . . . 2422
Table 573. SPI wake-up and interrupt requests . . . 2423
Table 574. Bitfields usable in PCM/I2S mode . . . 2425
Table 575. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 2434
Table 576. Serial data line swapping . . . 2434
Table 577. CLKGEN programming examples for usual I2S frequencies . . . 2438
Table 578. I2S interrupt requests . . . 2447
Table 579. SPI register map and reset values . . . 2465
Table 580. STM32H5 SAI features . . . 2468
Table 581. SAI internal input/output signals . . . 2470
Table 582. SAI input/output pins . . . 2470
Table 583. External synchronization selection . . . 2473
Table 584. MCLK_x activation conditions . . . 2478
Table 585. Clock generator programming examples . . . 2481
Table 586. SAI_A configuration for TDM mode . . . 2488
Table 587. TDM frame configuration examples . . . 2490
Table 588. SOPD pattern . . . 2494
Table 589. Parity bit calculation . . . 2494
Table 590. Audio sampling frequency versus symbol rates . . . 2495
Table 591. SAI interrupt sources . . . 2504
Table 592. SAI register map and reset values . . . 2533
Table 593. CAN subsystem I/O signals . . . 2539
Table 594. CAN subsystem I/O pins . . . 2539
Table 595. DLC coding in FDCAN . . . 2543
Table 596. Possible configurations for frame transmission . . . 2557
Table 597. Rx FIFO element . . . 2560
Table 598. Rx FIFO element description . . . 2560
Table 599. Tx buffer and FIFO element . . . 2562
Table 600. Tx buffer element description . . . 2562
Table 601. Tx event FIFO element . . . 2564
Table 602. Tx event FIFO element description . . . 2564
Table 603. Standard message ID filter element . . . 2565
Table 604. Standard message ID filter element field description . . . 2566
Table 605. Extended message ID filter element . . . 2566
Table 606. Extended message ID filter element field description . . . 2567
Table 607. FDCAN register map and reset values . . . 2597
Table 608. STM32H5 USB implementation . . . 2601
Table 609. USB input/output pins . . . 2602
Table 610. Double-buffering buffer flag definition . . . 2614
Table 611. Bulk double-buffering memory buffers usage (Device mode) . . . 2614
Table 612. Bulk double-buffering memory buffers usage (Host mode) . . . 2616
Table 613. Isochronous memory buffers usage . . . 2617

Table 614. Isochronous memory buffers usage . . . 2618
Table 615. Resume event detection . . . 2620
Table 616. Resume event detection for host . . . 2621
Table 617. Reception status encoding . . . 2639
Table 618. Endpoint/channel type encoding . . . 2639
Table 619. Endpoint/channel kind meaning . . . 2639
Table 620. Transmission status encoding . . . 2639
Table 621. USB register map and reset values . . . 2641
Table 622. Definition of allocated buffer memory . . . 2644
Table 623. USBSRAM register map and reset values . . . 2647
Table 624. UCPD implementation . . . 2649
Table 625. UCPD software trim data . . . 2649
Table 626. UCPD signals on pins . . . 2650
Table 627. UCPD internal signals . . . 2651
Table 628. 4b5b symbol encoding table . . . 2653
Table 629. Ordered sets . . . 2654
Table 630. Validation of ordered sets . . . 2654
Table 631. Data size . . . 2655
Table 632. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx . . . 2663
Table 633. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2) . . . 2665
Table 634. Effect of low power modes on the UCPD . . . 2667
Table 635. UCPD interrupt requests . . . 2668
Table 636. UCPD register map and reset values . . . 2684
Table 637. Ethernet peripheral pins . . . 2691
Table 638. Ethernet internal input/output signals . . . 2692
Table 639. Priority scheme for Tx DMA and Rx DMA . . . 2701
Table 640. Double VLAN processing features in Tx path . . . 2707
Table 641. Double VLAN processing in Rx path . . . 2708
Table 642. VLAN insertion or replacement based on VLTI bit . . . 2709
Table 643. Destination address filtering . . . 2712
Table 644. Source address filtering . . . 2713
Table 645. VLAN match status . . . 2714
Table 646. Ordinary clock: PTP messages for snapshot . . . 2717
Table 647. End-to-end transparent clock: PTP messages for snapshot . . . 2718
Table 648. Peer-to-peer transparent clock: PTP messages for snapshot . . . 2719
Table 649. Egress and ingress latency for PHY interfaces . . . 2722
Table 650. Minimum PTP clock frequency example . . . 2723
Table 651. Message format defined in IEEE 1588-2008 . . . 2724
Table 652. Message format defined in IEEE 1588-2008 . . . 2724
Table 653. IPv6-UDP PTP packet fields required for control and status . . . 2725
Table 654. Ethernet PTP packet fields required for control and status . . . 2726
Table 655. Timestamp Snapshot Dependency on ETH_MACTSCR bits . . . 2728
Table 656. PTP message generation criteria . . . 2734
Table 657. Common PTP message header fields . . . 2736
Table 658. MAC Transmit PTP mode and one-step timestamping operation . . . 2739
Table 659. Transmit checksum offload engine functions for different packet types . . . 2744
Table 660. Receive checksum offload engine functions for different packet types . . . 2746
Table 661. TSO: TCP and IP header fields . . . 2750
Table 662. Pause packet fields . . . 2755
Table 663. Tx MAC flow control . . . 2756
Table 664. Rx MAC flow control . . . 2756
Table 665. Size of the maximum receive packet . . . 2759

Table 666. MCD clock selection . . . 2762
Table 667. MDIO Clause 45 frame structure . . . 2763
Table 668. MDIO Clause 22 frame structure . . . 2764
Table 669. Remote wake-up packet filter register . . . 2775
Table 670. Description of the remote wake-up filter fields . . . 2775
Table 671. Remote wake-up packet and PMT interrupt generation . . . 2777
Table 672. Transfer complete interrupt behavior . . . 2785
Table 673. TDES0 normal descriptor (read format) . . . 2807
Table 674. TDES1 normal descriptor (read format) . . . 2808
Table 675. TDES2 normal descriptor (read format) . . . 2808
Table 676. TDES3 normal descriptor (read format) . . . 2809
Table 677. TDES0 normal descriptor (write-back format) . . . 2812
Table 678. TDES1 normal descriptor (write-back format) . . . 2812
Table 679. TDES2 normal descriptor (write-back format) . . . 2813
Table 680. TDES3 normal descriptor (write-back format) . . . 2813
Table 681. TDES0 context descriptor . . . 2816
Table 682. TDES1 context descriptor . . . 2817
Table 683. TDES2 context descriptor . . . 2817
Table 684. TDES3 context descriptor . . . 2817
Table 685. RDES0 normal descriptor (read format) . . . 2821
Table 686. RDES1 normal descriptor (read format) . . . 2821
Table 687. RDES2 normal descriptor (read format) . . . 2821
Table 688. RDES3 normal descriptor (read format) . . . 2821
Table 689. RDES0 normal descriptor (write-back format) . . . 2823
Table 690. RDES1 normal descriptor (write-back format) . . . 2823
Table 691. RDES2 normal descriptor (write-back format) . . . 2825
Table 692. RDES3 normal descriptor (write-back format) . . . 2827
Table 693. RDES0 context descriptor . . . 2830
Table 694. RDES1 context descriptor . . . 2831
Table 695. RDES2 context descriptor . . . 2831
Table 696. RDES3 context descriptor . . . 2831
Table 697. ETH_DMA common register map and reset values . . . 2853
Table 698. ETH_DMA_CH register map and reset values . . . 2853
Table 699. ETH_MTL register map and reset values . . . 2866
Table 700. Giant Packet Status based on S2KP and JE Bits . . . 2872
Table 701. Packet Length based on the CST and ACS bits . . . 2872
Table 702. Ethernet MAC register map and reset values . . . 2953
Table 703. HDMI pin . . . 2965
Table 704. HDMI-CEC internal input/output signals . . . 2965
Table 705. Error handling timing parameters . . . 2971
Table 706. TXERR timing parameters . . . 2972
Table 707. HDMI-CEC interrupts . . . 2973
Table 708. HDMI-CEC register map and reset values . . . 2981
Table 709. JTAG/Serial-wire debug port pins . . . 2983
Table 710. Trace port pins . . . 2983
Table 711. Single-wire trace port pins . . . 2984
Table 712. Authentication signal states with TrustZone enabled (TZEN = 0xB4) . . . 2986
Table 713. Authentication signal states with TrustZone disabled (TZEN = 0xC3) . . . 2986
Table 714. Life cycle state and debug states . . . 2986
Table 715. Definition of data to provision . . . 2989
Table 716. Permission mask (Endianness: Little Endian) . . . 2989
Table 717. JTAG-DP data registers . . . 2992

Table 718. Packet request . . . 2994
Table 719. ACK response . . . 2994
Table 720. Data transfer . . . 2994
Table 721. Debug port registers . . . 2995
Table 722. Debug port register map and reset values . . . 3001
Table 723. MEM-AP registers . . . 3003
Table 724. Access port register map and reset values . . . 3009
Table 725. System ROM table . . . 3010
Table 726. MCU ROM table . . . 3011
Table 727. Processor ROM table . . . 3011
Table 728. System ROM table register map and reset values . . . 3017
Table 729. MCU ROM table register map and reset values . . . 3022
Table 730. CPU ROM table register map and reset values . . . 3028
Table 731. DWT register map and reset values . . . 3043
Table 732. ITM register map and reset values . . . 3054
Table 733. BPU register map and reset values . . . 3062
Table 734. ETM register map and reset values . . . 3087
Table 735. TPIU register map and reset values . . . 3101
Table 736. CTI inputs . . . 3103
Table 737. CTI outputs . . . 3103
Table 738. CTI register map and reset values . . . 3114
Table 739. Peripheral clock freeze control bits . . . 3117
Table 740. Peripheral behaviour in debug mode . . . 3118
Table 741. DBGMCU register map and reset values . . . 3132
Table 742. Document revision history . . . 3140

List of figures

Figure 1. System architecture . . . 107
Figure 2. Memory map based on IDAU mapping (STM32H562/563/573xx devices) . . . 116
Figure 3. Memory map based on IDAU mapping (STM32H523/533xx devices) . . . 117
Figure 4. Secure/nonsecure partitioning using TrustZone technology . . . 133
Figure 5. Sharing memory map between CPU in secure and nonsecure state . . . 135
Figure 6. Secure world transition and memory partitioning . . . 135
Figure 7. Global TrustZone framework and TrustZone awareness . . . 137
Figure 8. Flash memory TrustZone protections . . . 141
Figure 9. Flash memory secure HDP area . . . 148
Figure 10. Key management principle . . . 156
Figure 11. Device life cycle security . . . 159
Figure 12. PRODUCT_STATEs (simplified TrustZone activated view) . . . 162
Figure 13. PRODUCT_STATEs (full TrustZone activated view) . . . 163
Figure 14. Collaborative development principle . . . 165
Figure 15. External flash memory protection using SFI . . . 167
Figure 16. GTZC in Armv8-M subsystem block diagram . . . 174
Figure 17. GTZC block diagram . . . 177
Figure 18. Watermark memory protection controller (region x/subregions A and B) . . . 178
Figure 19. MPCBB block diagram . . . 179
Figure 20. Memory map: SRAM1, SRAM2/3 with ECC (STM32H562/72/73xx devices) . . . 234
Figure 21. Memory map: SRAM1/3, SRAM2 with ECC (STM32H523/33xx devices) . . . 235
Figure 22. FLASH block diagram (simplified) . . . 247
Figure 23. Embedded flash memory organization (2-Mbyte devices) . . . 249
Figure 24. Embedded flash memory usage . . . 250
Figure 25. Flash high-cycle data memory map on 2-Mbyte devices . . . 263
Figure 26. Flash high-cycle data memory map on 1-Mbyte devices . . . 263
Figure 27. Flash high-cycle data memory map on 512-Kbyte devices . . . 264
Figure 28. Flash high-cycle data memory map on 256-Kbyte devices . . . 264
Figure 29. Flash bank swapping sequence . . . 267
Figure 30. OBK protection checks . . . 279
Figure 31. Key writing flow . . . 280
Figure 32. Swap workflow . . . 281
Figure 33. HDP in user flash memory (STM32H563/573xx devices) . . . 286
Figure 34. Protection attributes in case of bank swap illustration . . . 291
Figure 35. ICACHE block diagram . . . 368
Figure 36. ICACHE TAG and data memories functional view . . . 370
Figure 37. ICACHE remapping address mechanism . . . 373
Figure 38. DCACHE block diagram . . . 385
Figure 39. DCACHE TAG and data memories functional view . . . 388
Figure 40. Power supply with SMPS (STM32H563/573xx devices only) . . . 404
Figure 41. Power supply with LDO . . . 405
Figure 42. System supply configurations . . . 407
Figure 43. Power-on (POR) / power-down (PDR) reset waveform . . . 412
Figure 44. BOR thresholds . . . 413
Figure 45. PVD thresholds . . . 414
Figure 46. AVD thresholds . . . 415
Figure 47. VBAT thresholds . . . 416
Figure 48. Temperature thresholds . . . 417

Figure 49. Dynamic voltage scaling in Run mode . . . 418
Figure 50. I/O states in Standby mode . . . 428
Figure 51. Simplified diagram of the reset circuit . . . 453
Figure 52. Clock tree . . . 456
Figure 53. HSE/LSE clock sources . . . 457
Figure 54. CSI calibration flow . . . 460
Figure 55. PLL block diagram . . . 461
Figure 56. PLLs initialization flow . . . 464
Figure 57. CRS block diagram . . . 564
Figure 58. CRS counter behavior . . . 566
Figure 59. Structure of 3- or 5-V tolerant GPIO (TT or FT) . . . 576
Figure 60. Input floating/pull-up/pull-down configurations . . . 580
Figure 61. Output configuration . . . 581
Figure 62. Alternate function configuration . . . 582
Figure 63. High-impedance analog configuration . . . 582
Figure 64. SBS block diagram . . . 597
Figure 65. Compensation cell management . . . 599
Figure 66. Compensation cell usage . . . 600
Figure 67. SBS boot control . . . 601
Figure 68. SBS debug control . . . 604
Figure 69. SBS hardware secure storage control . . . 606
Figure 70. GPDMA block diagram . . . 641
Figure 71. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) . . . 642
Figure 72. GPDMA channel suspend and resume sequence . . . 643
Figure 73. GPDMA channel abort and restart sequence . . . 644
Figure 74. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 645
Figure 75. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 646
Figure 76. GPDMA dynamic linked-list data structure of a linear addressing channel x . . . 647
Figure 77. GPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 647
Figure 78. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) . . . 649
Figure 79. Inserting a LLIn with an auxiliary GPDMA channel y . . . 651
Figure 80. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) . . . 653
Figure 81. Building LLIn+1: GPDMA dynamic linked-lists in link step mode . . . 654
Figure 82. Replace with a new LLIn' in register file in link step mode . . . 655
Figure 83. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 656
Figure 84. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 657
Figure 85. GPDMA channel execution and linked-list programming . . . 659
Figure 86. Programmed 2D addressing . . . 662
Figure 87. GPDMA arbitration policy . . . 669
Figure 88. Trigger hit, memorization, and overrun waveform . . . 672
Figure 89. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 673
Figure 90. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 674
Figure 91. EXTI block diagram . . . 723
Figure 92. Configurable event trigger logic CPU wake-up . . . 726

Figure 93. EXTI direct events . . . 727
Figure 94. EXTI mux GPIO selection . . . 728
Figure 95. CRC calculation unit block diagram . . . 757
Figure 96. CORDIC convergence for trigonometric functions . . . 771
Figure 97. CORDIC convergence for hyperbolic functions . . . 772
Figure 98. CORDIC convergence for square root . . . 773
Figure 99. Block diagram . . . 782
Figure 100. Input buffer areas . . . 784
Figure 101. Circular input buffer . . . 785
Figure 102. Circular input buffer operation . . . 786
Figure 103. Circular output buffer . . . 787
Figure 104. Circular output buffer operation . . . 788
Figure 105. FIR filter structure . . . 790
Figure 106. IIR filter structure (direct form 1) . . . 792
Figure 107. X1 buffer initialization . . . 797
Figure 108. Filtering example 1 . . . 798
Figure 109. Filtering example 2 . . . 799
Figure 110. FMC block diagram . . . 811
Figure 111. FMC memory banks . . . 813
Figure 112. Mode 1 read access waveforms . . . 823
Figure 113. Mode 1 write access waveforms . . . 823
Figure 114. Mode A read access waveforms . . . 825
Figure 115. Mode A write access waveforms . . . 825
Figure 116. Mode 2 and mode B read access waveforms . . . 827
Figure 117. Mode 2 write access waveforms . . . 828
Figure 118. Mode B write access waveforms . . . 828
Figure 119. Mode C read access waveforms . . . 830
Figure 120. Mode C write access waveforms . . . 831
Figure 121. Mode D read access waveforms . . . 833
Figure 122. Mode D write access waveforms . . . 834
Figure 123. Muxed read access waveforms . . . 836
Figure 124. Muxed write access waveforms . . . 837
Figure 125. Asynchronous wait during a read access waveforms . . . 839
Figure 126. Asynchronous wait during a write access waveforms . . . 840
Figure 127. Wait configuration waveforms . . . 842
Figure 128. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) . . . 843
Figure 129. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) . . . 845
Figure 130. NAND flash controller waveforms for common memory access . . . 857
Figure 131. Access to non ‘CE don’t care’ NAND-flash . . . 859
Figure 132. Burst write SDRAM access waveforms . . . 868
Figure 133. Burst read SDRAM access . . . 869
Figure 134. Logic diagram of Read access with RBURST bit set (CAS=1, RPIPE=0) . . . 870
Figure 135. Read access crossing row boundary . . . 872
Figure 136. Write access crossing row boundary . . . 872
Figure 137. Self-refresh mode . . . 874
Figure 138. Power-down mode . . . 875
Figure 139. OCTOSPI block diagram in octal configuration . . . 887
Figure 140. OCTOSPI block diagram in quad configuration . . . 887
Figure 141. OCTOSPI block diagram in dual-quad configuration . . . 888
Figure 142. SDR read command in octal configuration . . . 889
Figure 143. DTR read in octal-SPI mode with DQS (Macronix mode) example . . . 892
Figure 144. SDR write command in octo-SPI mode example . . . 894

Figure 145. DTR write in octal-SPI mode (Macronix mode) example . . . 895
Figure 146. Example of HyperBus read operation . . . 896
Figure 147. HyperBus write operation with initial latency . . . 898
Figure 148. HyperBus read operation with additional latency . . . 898
Figure 149. HyperBus write operation with additional latency . . . 899
Figure 150. HyperBus write operation with no latency (register write) . . . 899
Figure 151. HyperBus read operation page crossing with latency . . . 900
Figure 152. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access . . . 906
Figure 153. OctaRAM read operation with reverse data ordering D1/D0 . . . 907
Figure 154. NCS when CKMODE = 0 (T = CLK period) . . . 912
Figure 155. NCS when CKMODE = 1 in SDR mode (T = CLK period) . . . 913
Figure 156. NCS when CKMODE = 1 in DTR mode (T = CLK period) . . . 913
Figure 157. NCS when CKMODE = 1 with an abort (T = CLK period) . . . 913
Figure 158. SDMMC “no response” and “no data” operations . . . 942
Figure 159. SDMMC (multiple) block read operation . . . 942
Figure 160. SDMMC (multiple) block write operation . . . 943
Figure 161. SDMMC (sequential) stream read operation . . . 943
Figure 162. SDMMC (sequential) stream write operation . . . 943
Figure 163. SDMMC block diagram . . . 945
Figure 164. SDMMC Command and data phase relation . . . 947
Figure 165. Control unit . . . 949
Figure 166. Command/response path . . . 950
Figure 167. Command path state machine (CPSM) . . . 951
Figure 168. Data path . . . 957
Figure 169. DDR mode data packet clocking . . . 958
Figure 170. DDR mode CRC status / boot acknowledgment clocking . . . 958
Figure 171. Data path state machine (DPSM) . . . 959
Figure 172. CLKMUX unit . . . 970
Figure 173. Linked list structures . . . 972
Figure 174. Asynchronous interrupt generation . . . 975
Figure 175. Synchronous interrupt period data read . . . 976
Figure 176. Synchronous interrupt period data write . . . 976
Figure 177. Asynchronous interrupt period data read . . . 977
Figure 178. Asynchronous interrupt period data write . . . 978
Figure 179. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25 . . . 981
Figure 180. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104 . . . 981
Figure 181. Read Wait with SDMMC_CK < 50 MHz . . . 982
Figure 182. Read Wait with SDMMC_CK > 50 MHz . . . 982
Figure 183. CMD12 stream timing . . . 985
Figure 184. CMD5 Sleep Awake procedure . . . 987
Figure 185. Normal boot mode operation . . . 989
Figure 186. Alternative boot mode operation . . . 990
Figure 187. Command response R1b busy signaling . . . 991
Figure 188. SDMMC state control . . . 992
Figure 189. Card cycle power / power up diagram . . . 993
Figure 190. CMD11 signal voltage switch sequence . . . 994
Figure 191. Voltage switch transceiver typical application . . . 996
Figure 192. DLYB block diagram . . . 1024
Figure 193. ADC block diagram . . . 1032
Figure 194. ADC clock scheme . . . 1037
Figure 195. ADC1 connectivity . . . 1038
Figure 196. ADC2 connectivity . . . 1039

Figure 197. ADC calibration . . . 1042
Figure 198. Updating the ADC calibration factor . . . 1043
Figure 199. Mixing single-ended and differential channels . . . 1044
Figure 200. Enabling / disabling the ADC . . . 1045
Figure 201. Bulb mode timing diagram . . . 1048
Figure 202. Analog-to-digital conversion time . . . 1051
Figure 203. Stopping ongoing regular conversions . . . 1052
Figure 204. Stopping ongoing regular and injected conversions . . . 1053
Figure 205. Triggers shared between ADC master and slave . . . 1054
Figure 206. Injected conversion latency . . . 1056
Figure 207. Example of ADC_JSQR queue of context (sequence change) . . . 1059
Figure 208. Example of ADC_JSQR queue of context (trigger change) . . . 1059
Figure 209. Example of ADC_JSQR queue of context with overflow before conversion . . . 1060
Figure 210. Example of ADC_JSQR queue of context with overflow during conversion . . . 1060
Figure 211. Example of ADC_JSQR queue of context with empty queue (case JQM = 0) . . . 1061
Figure 212. Example of ADC_JSQR queue of context with empty queue (JQM = 1) . . . 1062
Figure 213. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion . . . 1062
Figure 214. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs . . . 1063
Figure 215. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion . . . 1063
Figure 216. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1) . . . 1064
Figure 217. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0) . . . 1064
Figure 218. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1) . . . 1065
Figure 219. Single conversions of a sequence, software trigger . . . 1067
Figure 220. Continuous conversion of a sequence, software trigger . . . 1067
Figure 221. Single conversions of a sequence, hardware trigger . . . 1068
Figure 222. Continuous conversions of a sequence, hardware trigger . . . 1068
Figure 223. Right alignment (offset disabled, unsigned value) . . . 1070
Figure 224. Right alignment (offset enabled, signed value) . . . 1071
Figure 225. Left alignment (offset disabled, unsigned value) . . . 1071
Figure 226. Left alignment (offset enabled, signed value) . . . 1072
Figure 227. Example of overrun (OVRMOD = 0) . . . 1073
Figure 228. Example of overrun (OVRMOD = 1) . . . 1074
Figure 229. AUTODLY = 1, regular conversion in continuous mode, software trigger . . . 1077
Figure 230. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 1077
Figure 231. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . 1078
Figure 232. AUTODLY = 1, regular continuous conversions interrupted by injected conversions . . . 1079
Figure 233. AUTODLY = 1 in auto- injected mode (JAUTO = 1) . . . 1079
Figure 234. Analog watchdog guarded area . . . 1080
Figure 235. ADCy_AWDx_OUT signal generation (on all regular channels) . . . 1082
Figure 236. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software) . . . 1083
Figure 237. ADCy_AWDx_OUT signal generation (on a single regular channel) . . . 1083
Figure 238. ADCy_AWDx_OUT signal generation (on all injected channels) . . . 1083
Figure 239. 20-bit to 16-bit result truncation . . . 1085
Figure 240. Numerical example with 5-bit shift and rounding . . . 1085
Figure 241. Triggered regular oversampling mode (TROVS bit = 1) . . . 1087
Figure 242. Regular oversampling modes (4x ratio) . . . 1088

Figure 243. Regular and injected oversampling modes used simultaneously . . . 1089
Figure 244. Triggered regular oversampling with injection . . . 1089
Figure 245. Oversampling in auto-injected mode . . . 1090
Figure 246. Dual ADC block diagram(1) . . . 1092
Figure 247. Injected simultaneous mode on 4 channels: dual ADC mode . . . 1093
Figure 248. Regular simultaneous mode on 16 channels: dual ADC mode . . . 1095
Figure 249. Interleaved mode on one channel in continuous conversion mode: dual ADC mode . . . 1096
Figure 250. Interleaved mode on 1 channel in single conversion mode: dual ADC mode . . . 1097
Figure 251. Interleaved conversion with injection . . . 1097
Figure 252. Alternate trigger: injected group of each ADC . . . 1098
Figure 253. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 1099
Figure 254. Alternate + regular simultaneous . . . 1100
Figure 255. Case of trigger occurring during injected conversion . . . 1100
Figure 256. Interleaved single channel CH0 with injected sequence CH11, CH12 . . . 1101
Figure 257. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first . . . 1101
Figure 258. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first . . . 1101
Figure 259. DMA Requests in regular simultaneous mode when MDMA = 0b00 . . . 1102
Figure 260. DMA requests in regular simultaneous mode when MDMA = 0b10 . . . 1103
Figure 261. DMA requests in interleaved mode when MDMA = 0b10 . . . 1103
Figure 262. Temperature sensor channel block diagram . . . 1105
Figure 263. VBAT channel block diagram . . . 1107
Figure 264. VREFINT channel block diagram . . . 1107
Figure 265. Temperature sensor functional block diagram . . . 1151
Figure 266. Method for low REF_CLK frequencies . . . 1153
Figure 267. Method for high REF_CLK frequencies . . . 1153
Figure 268. Temperature sensor sequence . . . 1156
Figure 269. Dual-channel DAC block diagram . . . 1169
Figure 270. Data registers in single DAC channel mode . . . 1172
Figure 271. Data registers in dual DAC channel mode . . . 1172
Figure 272. Timing diagram for conversion with trigger disabled TEN = 0 . . . 1174
Figure 273. DAC LFSR register calculation algorithm . . . 1176
Figure 274. DAC conversion (SW trigger enabled) with LFSR wave generation . . . 1177
Figure 275. DAC triangle wave generation . . . 1177
Figure 276. DAC conversion (SW trigger enabled) with triangle wave generation . . . 1178
Figure 277. DAC sample and hold mode phase diagram . . . 1180
Figure 278. VREFBUF block diagram . . . 1207
Figure 279. DCMI block diagram . . . 1212
Figure 280. DCMI signal waveforms . . . 1213
Figure 281. Timing diagram . . . 1215
Figure 282. Frame capture waveforms in snapshot mode . . . 1217
Figure 283. Frame capture waveforms in continuous grab mode . . . 1218
Figure 284. Coordinates and size of the window after cropping . . . 1218
Figure 285. Data capture waveforms . . . 1219
Figure 286. Pixel raster scan order . . . 1220
Figure 287. PSSI block diagram . . . 1234
Figure 288. Top-level block diagram . . . 1234
Figure 289. Data enable in receive mode waveform diagram (CKPOL=0) . . . 1238
Figure 290. Data enable waveform diagram in transmit mode (CKPOL=1) . . . 1238
Figure 291. Ready in receive mode waveform diagram (CKPOL=0) . . . 1239
Figure 292. Bidirectional PSSI_DE/PSSI_RDY waveform . . . 1240

Figure 293. Bidirectional PSSI_DE/PSSI_RDY connection diagram 1240
Figure 294. RNG block diagram 1248
Figure 295. NIST SP800-90B entropy source model 1249
Figure 296. RNG initialization overview 1252
Figure 297. AES block diagram 1266
Figure 298. Encryption/ decryption typical usage 1267
Figure 299. Typical operation with authentication 1270
Figure 300. Example of suspend mode management 1271
Figure 301. ECB encryption 1272
Figure 302. ECB decryption 1272
Figure 303. CBC encryption 1272
Figure 304. CBC decryption 1273
Figure 305. Message construction in CTR mode 1275
Figure 306. CTR encryption 1276
Figure 307. Message construction in GCM 1277
Figure 308. GCM authenticated encryption 1279
Figure 309. Message construction in GMAC mode 1282
Figure 310. GMAC authentication mode 1283
Figure 311. Message construction in CCM mode 1284
Figure 312. CCM mode authenticated encryption 1285
Figure 313. 128-bit block construction according to the data type 1290
Figure 314. SAES block diagram 1312
Figure 315. Encryption/ decryption typical usage 1314
Figure 316. Typical operation with authentication 1316
Figure 317. Example of suspend mode management 1317
Figure 318. ECB encryption 1318
Figure 319. ECB decryption 1318
Figure 320. CBC encryption 1319
Figure 321. CBC decryption 1319
Figure 322. Message construction in CTR mode 1322
Figure 323. CTR encryption 1323
Figure 324. Message construction in GCM 1324
Figure 325. GCM authenticated encryption 1326
Figure 326. Message construction in GMAC mode 1329
Figure 327. GMAC authentication mode 1329
Figure 328. Message construction in CCM mode 1330
Figure 329. CCM mode authenticated encryption 1332
Figure 330. Operation with wrapped keys for SAES in ECB and CBC modes 1335
Figure 331. Operation with wrapped keys for SAES in CTR mode 1338
Figure 332. Usage of Shared-key mode 1339
Figure 333. 128-bit block construction according to the data type 1342
Figure 334. Key protection mechanisms 1344
Figure 335. HASH block diagram 1366
Figure 336. Message data swapping feature 1368
Figure 337. HASH suspend/resume mechanism 1374
Figure 338. PKA block diagram 1388
Figure 339. OTFDEC block diagram 1421
Figure 340. Typical OTFDEC use in a SoC 1422
Figure 341. AES CTR decryption flow 1423
Figure 342. OTFDEC flow control overview (dual burst read request) 1424
Figure 343. Advanced-control timer block diagram 1444
Figure 344. Counter timing diagram with prescaler division change from 1 to 2 1450

Figure 345. Counter timing diagram with prescaler division change from 1 to 4 1450
Figure 346. Counter timing diagram, internal clock divided by 1 1452
Figure 347. Counter timing diagram, internal clock divided by 2 1452
Figure 348. Counter timing diagram, internal clock divided by 4 1453
Figure 349. Counter timing diagram, internal clock divided by N. 1453
Figure 350. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). 1454
Figure 351. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). 1455
Figure 352. Counter timing diagram, internal clock divided by 1 1456
Figure 353. Counter timing diagram, internal clock divided by 2 1457
Figure 354. Counter timing diagram, internal clock divided by 4 1457
Figure 355. Counter timing diagram, internal clock divided by N. 1458
Figure 356. Counter timing diagram, update event when repetition counter is not used. 1458
Figure 357. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 1460
Figure 358. Counter timing diagram, internal clock divided by 2 1460
Figure 359. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 1461
Figure 360. Counter timing diagram, internal clock divided by N. 1461
Figure 361. Counter timing diagram, update event with ARPE = 1 (counter underflow) 1462
Figure 362. Counter timing diagram, Update event with ARPE = 1 (counter overflow). 1463
Figure 363. Update rate examples depending on mode and TIMx_RCR register settings 1464
Figure 365. Control circuit in normal mode, internal clock divided by 1. 1466
Figure 366. tim_ti2 external clock connection example 1466
Figure 367. Control circuit in external clock mode 1 1467
Figure 368. External trigger input block 1468
Figure 369. Control circuit in external clock mode 2 1469
Figure 370. Capture/compare channel (example: channel 1 input stage) 1469
Figure 371. Capture/compare channel 1 main circuit 1470
Figure 372. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) 1471
Figure 373. Output stage of capture/compare channel (channel 5, idem ch. 6) 1471
Figure 374. PWM input mode timing 1474
Figure 375. Output compare mode, toggle on tim_oc1 1476
Figure 376. Edge-aligned PWM waveforms (ARR = 8) 1477
Figure 377. Center-aligned PWM waveforms (ARR = 8). 1478
Figure 378. Dithering principle 1479
Figure 379. Data format and register coding in dithering mode. 1480
Figure 380. PWM resolution vs frequency 1481
Figure 381. PWM dithering pattern 1482
Figure 382. Dithering effect on duty cycle in center-aligned PWM mode 1483
Figure 383. Generation of 2 phase-shifted PWM signals with 50% duty cycle 1485
Figure 384. Combined PWM mode on channel 1 and 3 1486
Figure 385. 3-phase combined PWM signals with multiple trigger pulses per period 1487
Figure 386. Complementary output with symmetrical dead-time insertion 1488
Figure 387. Asymmetrical deadtime 1489
Figure 388. Dead-time waveforms with delay greater than the negative pulse 1489
Figure 389. Dead-time waveforms with delay greater than the positive pulse. 1489
Figure 390. Break and Break2 circuitry overview 1492
Figure 391. Various output behavior in response to a break event on tim_brk (OSSI = 1) 1494
Figure 392. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) 1495
Figure 393. PWM output state following tim_brk assertion (OSSI = 0) 1496
Figure 394. Output redirection (tim_brk2 request not represented). 1497
Figure 395. tim_ocref_clr input selection multiplexer. 1498

Figure 396. Clearing TIMx tim_ocxref . . . 1499
Figure 397. 6-step generation, COM example (OSSR = 1) . . . 1500
Figure 398. Example of one pulse mode. . . . 1501
Figure 399. Retriggerable one-pulse mode . . . 1502
Figure 400. Pulse generator circuitry . . . 1503
Figure 401. Pulse generation on compare event, for edge-aligned and encoder modes . . . 1504
Figure 402. Extended pulsewidth in case of concurrent triggers . . . 1505
Figure 403. Example of counter operation in encoder interface mode. . . . 1507
Figure 404. Example of encoder interface mode with tim_ti1fp1 polarity inverted. . . . 1507
Figure 405. Quadrature encoder counting modes . . . 1508
Figure 406. Direction plus clock encoder mode . . . 1509
Figure 407. Directional clock encoder mode (CC1P = CC2P = 0) . . . 1509
Figure 408. Directional clock encoder mode (CC1P = CC2P = 1) . . . 1510
Figure 409. Index gating options . . . 1511
Figure 410. Jittered Index signals . . . 1511
Figure 411. Index generation for IPOS[1:0] = 11 . . . 1512
Figure 412. Counter reading with index gated on channel A (IPOS[1:0] = 11) . . . 1512
Figure 413. Counter reading with index ungated (IPOS[1:0] = 00) . . . 1513
Figure 414. Counter reading with index gated on channel A and B. . . . 1513
Figure 415. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) . . . 1514
Figure 416. Counter reset Narrow index pulse (closer view, ARR = 0x07) . . . 1515
Figure 417. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) . . . 1516
Figure 418. Directional index sensitivity . . . 1516
Figure 419. Counter reset as function of FIDX bit setting . . . 1517
Figure 420. Index blanking . . . 1517
Figure 421. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 1518
Figure 422. Index behavior in directional clock mode, IPOS[0] = 1 . . . 1518
Figure 423. State diagram for quadrature encoded signals . . . 1519
Figure 424. Up-counting encoder error detection . . . 1520
Figure 425. Down-counting encode error detection . . . 1521
Figure 426. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 1522
Figure 427. Measuring time interval between edges on three signals . . . 1523
Figure 428. Example of Hall sensor interface . . . 1525
Figure 429. Control circuit in reset mode . . . 1526
Figure 430. Control circuit in Gated mode . . . 1527
Figure 431. Control circuit in trigger mode . . . 1528
Figure 432. Control circuit in external clock mode 2 + trigger mode . . . 1529
Figure 433. General-purpose timer block diagram . . . 1587
Figure 434. Counter timing diagram with prescaler division change from 1 to 2 . . . 1592
Figure 435. Counter timing diagram with prescaler division change from 1 to 4 . . . 1593
Figure 436. Counter timing diagram, internal clock divided by 1 . . . 1594
Figure 437. Counter timing diagram, internal clock divided by 2 . . . 1594
Figure 438. Counter timing diagram, internal clock divided by 4 . . . 1595
Figure 439. Counter timing diagram, internal clock divided by N. . . . 1595
Figure 440. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded). . . 1596
Figure 441. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded). 1597
Figure 442. Counter timing diagram, internal clock divided by 1 . . . 1598
Figure 443. Counter timing diagram, internal clock divided by 2 . . . 1599
Figure 444. Counter timing diagram, internal clock divided by 4 . . . 1599
Figure 445. Counter timing diagram, internal clock divided by N. . . . 1600
Figure 446. Counter timing diagram, Update event . . . 1600
Figure 447. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 1602

Figure 448. Counter timing diagram, internal clock divided by 2 1602
Figure 449. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 1603
Figure 450. Counter timing diagram, internal clock divided by N. 1603
Figure 451. Counter timing diagram, Update event with ARPE = 1 (counter underflow). 1604
Figure 452. Counter timing diagram, Update event with ARPE = 1 (counter overflow). 1605
Figure 453. Control circuit in normal mode, internal clock divided by 1. 1606
Figure 454. tim_ti2 external clock connection example 1606
Figure 455. Control circuit in external clock mode 1 1607
Figure 456. External trigger input block 1608
Figure 457. Control circuit in external clock mode 2 1609
Figure 458. Capture/compare channel (example: channel 1 input stage) 1609
Figure 459. Capture/compare channel 1 main circuit 1610
Figure 460. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4). 1610
Figure 461. PWM input mode timing 1613
Figure 462. Output compare mode, toggle on tim_oc1 1615
Figure 463. Edge-aligned PWM waveforms (ARR = 8) 1616
Figure 464. Center-aligned PWM waveforms (ARR = 8). 1617
Figure 465. Dithering principle 1618
Figure 466. Data format and register coding in dithering mode. 1619
Figure 467. PWM resolution vs frequency (16-bit mode). 1620
Figure 468. PWM resolution vs frequency (32-bit mode). 1620
Figure 469. PWM dithering pattern 1621
Figure 470. Dithering effect on duty cycle in center-aligned PWM mode 1622
Figure 471. Generation of two phase-shifted PWM signals with 50% duty cycle 1624
Figure 472. Combined PWM mode on channels 1 and 3 1625
Figure 473. OCREF_CLR input selection multiplexer 1626
Figure 474. Clearing TIMx tim_ocxref 1626
Figure 475. Example of One-pulse mode 1627
Figure 476. Retriggerable one-pulse mode 1629
Figure 477. Pulse generator circuitry 1630
Figure 478. Pulse generation on compare event, for edge-aligned and encoder modes 1630
Figure 479. Extended pulse width in case of concurrent triggers 1631
Figure 480. Example of counter operation in encoder interface mode 1633
Figure 481. Example of encoder interface mode with tim_ti1fp1 polarity inverted 1633
Figure 482. Quadrature encoder counting modes 1634
Figure 483. Direction plus clock encoder mode. 1635
Figure 484. Directional clock encoder mode (CC1P = CC2P = 0). 1636
Figure 485. Directional clock encoder mode (CC1P = CC2P = 1). 1636
Figure 486. Index gating options 1638
Figure 487. Jittered Index signals 1638
Figure 488. Index generation for IPOS[1:0] = 11 1639
Figure 489. Counter reading with index gated on channel A (IPOS[1:0] = 11) 1639
Figure 490. Counter reading with index ungated (IPOS[1:0] = 00) 1640
Figure 491. Counter reading with index gated on channel A and B. 1640
Figure 492. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11). 1641
Figure 493. Counter reset Narrow index pulse (closer view, ARR = 0x07) 1642
Figure 494. Index behavior in x1 and x2 mode (IPOS[1:0] = 01). 1643
Figure 495. Directional index sensitivity. 1643
Figure 496. Counter reset as function of FIDX bit setting 1644
Figure 497. Index blanking. 1644
Figure 498. Index behavior in clock + direction mode, IPOS[0] = 1. 1645
Figure 499. Index behavior in directional clock mode, IPOS[0] = 1 1645

Figure 500. State diagram for quadrature encoded signals . . . 1646
Figure 501. Up-counting encoder error detection . . . 1647
Figure 502. Down-counting encode error detection . . . 1648
Figure 503. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 1649
Figure 504. Control circuit in reset mode . . . 1651
Figure 505. Control circuit in gated mode . . . 1652
Figure 506. Control circuit in trigger mode . . . 1652
Figure 507. Control circuit in external clock mode 2 + trigger mode . . . 1654
Figure 508. Master/Slave timer example . . . 1654
Figure 509. Master/slave connection example with 1 channel only timers . . . 1655
Figure 510. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 1656
Figure 511. Gating TIM_slv with Enable of TIM_mstr . . . 1657
Figure 512. Triggering TIM_slv with update of TIM_mstr. . . . 1658
Figure 513. Triggering TIM_slv with Enable of TIM_mstr . . . 1658
Figure 514. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 1659
Figure 515. Basic timer block diagram. . . . 1703
Figure 516. Control circuit in normal mode, internal clock divided by 1 . . . 1704
Figure 517. Counter timing diagram with prescaler division change from 1 to 2 . . . 1705
Figure 518. Counter timing diagram with prescaler division change from 1 to 4 . . . 1706
Figure 519. Counter timing diagram, internal clock divided by 1 . . . 1707
Figure 520. Counter timing diagram, internal clock divided by 2 . . . 1707
Figure 521. Counter timing diagram, internal clock divided by 4 . . . 1708
Figure 522. Counter timing diagram, internal clock divided by N. . . . 1708
Figure 523. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). . . . 1709
Figure 524. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). . . . 1710
Figure 525. Dithering principle . . . 1711
Figure 526. Data format and register coding in dithering mode . . . 1711
Figure 527. FCnt resolution vs frequency . . . 1712
Figure 528. PWM dithering pattern . . . 1712
Figure 529. General-purpose timer block diagram (TIM12) . . . 1723
Figure 530. General-purpose timer block diagram (TIM13/TIM14)(2) . . . 1724
Figure 531. Counter timing diagram with prescaler division change from 1 to 2 . . . 1728
Figure 532. Counter timing diagram with prescaler division change from 1 to 4 . . . 1728
Figure 533. Counter timing diagram, internal clock divided by 1 . . . 1729
Figure 534. Counter timing diagram, internal clock divided by 2 . . . 1730
Figure 535. Counter timing diagram, internal clock divided by 4 . . . 1730
Figure 536. Counter timing diagram, internal clock divided by N. . . . 1731
Figure 537. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). . . . 1731
Figure 538. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). . . . 1732
Figure 539. Control circuit in normal mode, internal clock divided by 1 . . . 1733
Figure 540. tim_ti2 external clock connection example . . . 1733
Figure 541. Control circuit in external clock mode 1 . . . 1734
Figure 542. Capture/compare channel 1 input stage (TIM13/TIM14) . . . 1735
Figure 543. Capture/compare channel 1 input stage (TIM12) . . . 1735
Figure 544. Capture/compare channel 1 main circuit . . . 1736
Figure 545. Output stage of capture/compare channel 1. . . . 1736
Figure 546. PWM input mode timing . . . 1738
Figure 547. Output compare mode, toggle on tim_oc1. . . . 1740

Figure 548. Edge-aligned PWM waveforms (ARR = 8) . . . 1741
Figure 549. Dithering principle . . . 1742
Figure 550. Data format and register coding in dithering mode . . . 1742
Figure 551. PWM resolution vs frequency . . . 1743
Figure 552. PWM dithering pattern . . . 1744
Figure 553. Combined PWM mode on channel 1 and 2 . . . 1746
Figure 554. Example of one pulse mode . . . 1747
Figure 555. Retriggerable one pulse mode . . . 1748
Figure 556. Measuring time interval between edges on 2 signals . . . 1749
Figure 557. Control circuit in reset mode . . . 1750
Figure 558. Control circuit in gated mode . . . 1751
Figure 559. Control circuit in trigger mode . . . 1751
Figure 560. TIM15 block diagram . . . 1786
Figure 561. TIM16/TIM17 block diagram(3) . . . 1787
Figure 562. Counter timing diagram with prescaler division change from 1 to 2 . . . 1791
Figure 563. Counter timing diagram with prescaler division change from 1 to 4 . . . 1792
Figure 564. Counter timing diagram, internal clock divided by 1 . . . 1793
Figure 565. Counter timing diagram, internal clock divided by 2 . . . 1794
Figure 566. Counter timing diagram, internal clock divided by 4 . . . 1794
Figure 567. Counter timing diagram, internal clock divided by N . . . 1795
Figure 568. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 1795
Figure 569. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 1796
Figure 570. Update rate examples depending on mode and TIMx_RCR register settings . . . 1797
Figure 571. Control circuit in normal mode, internal clock divided by 1 . . . 1798
Figure 572. tim_ti2 external clock connection example . . . 1798
Figure 573. Control circuit in external clock mode 1 . . . 1799
Figure 574. Capture/compare channel (example: channel 1 input stage) . . . 1800
Figure 575. Capture/compare channel 1 main circuit . . . 1800
Figure 576. Output stage of capture/compare channel (channel 1) . . . 1801
Figure 577. Output stage of capture/compare channel (channel 2 for TIM15) . . . 1801
Figure 578. PWM input mode timing . . . 1804
Figure 579. Output compare mode, toggle on tim_oc1 . . . 1806
Figure 580. Edge-aligned PWM waveforms (ARR = 8) . . . 1807
Figure 581. Dithering principle . . . 1808
Figure 582. Data format and register coding in dithering mode . . . 1808
Figure 583. PWM resolution vs frequency . . . 1809
Figure 584. PWM dithering pattern . . . 1810
Figure 585. Combined PWM mode on channel 1 and 2 . . . 1812
Figure 586. Complementary output with symmetrical dead-time insertion . . . 1813
Figure 587. Asymmetrical deadtime . . . 1814
Figure 588. Dead-time waveforms with delay greater than the negative pulse . . . 1814
Figure 589. Dead-time waveforms with delay greater than the positive pulse . . . 1814
Figure 590. Break circuitry overview . . . 1816
Figure 591. Output behavior in response to a break event on tim_brk . . . 1818
Figure 592. Output redirection . . . 1820
Figure 593. tim_ocref_clr input selection multiplexer . . . 1821
Figure 594. 6-step generation, COM example (OSSR = 1) . . . 1822
Figure 595. Example of one pulse mode . . . 1823
Figure 596. Retriggerable one pulse mode . . . 1825
Figure 597. Measuring time interval between edges on 2 signals . . . 1825

Figure 598. Control circuit in reset mode . . . 1826
Figure 599. Control circuit in gated mode . . . 1827
Figure 600. Control circuit in trigger mode . . . 1828
Figure 601. LPTIM1/2/3/5/6 block diagram(1) . . . 1892
Figure 602. LPTIM4 block diagram(1) . . . 1893
Figure 603. Glitch filter timing diagram . . . 1897
Figure 604. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 1899
Figure 605. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 1900
Figure 606. LPTIM output waveform, Continuous counting mode configuration . . . 1900
Figure 607. Waveform generation . . . 1902
Figure 608. Encoder mode counting sequence . . . 1906
Figure 609. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1) . . . 1907
Figure 610. Capture/compare input stage (channel 1) . . . 1908
Figure 611. Capture/compare output stage (channel 1) . . . 1908
Figure 612. Edge-aligned PWM mode (PRELOAD = 1) . . . 1910
Figure 613. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 1911
Figure 614. PWM mode with immediate update versus preloaded update . . . 1912
Figure 615. Independent watchdog block diagram . . . 1942
Figure 616. Reset timing due to timeout . . . 1944
Figure 617. Reset timing due to refresh in the not allowed area . . . 1945
Figure 618. Changing PR, RL, and performing a refresh(1) . . . 1946
Figure 619. Window comparator update(1) . . . 1947
Figure 620. Independent watchdog interrupt timing diagram . . . 1949
Figure 621. Early wake-up comparator update(1) . . . 1950
Figure 622. Watchdog block diagram . . . 1958
Figure 623. Window watchdog timing diagram . . . 1960
Figure 624. RTC block diagram . . . 1966
Figure 625. TAMP block diagram . . . 2024
Figure 626. Backup registers protection zones . . . 2029
Figure 627. Tamper sampling with precharge pulse . . . 2034
Figure 628. Low level detection with precharge and filtering . . . 2034
Figure 629. Active tamper filtering . . . 2036
Figure 630. Block diagram . . . 2073
Figure 631. I²C-bus protocol . . . 2075
Figure 632. Setup and hold timings . . . 2077
Figure 633. I2C initialization flow . . . 2079
Figure 634. Data reception . . . 2080
Figure 635. Data transmission . . . 2081
Figure 636. Target initialization flow . . . 2084
Figure 637. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 2086
Figure 638. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 2087
Figure 639. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 2088
Figure 640. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 2089
Figure 641. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 2090
Figure 642. Transfer bus diagrams for I2C target receiver (mandatory events only) . . . 2090
Figure 643. Controller clock generation . . . 2092
Figure 644. Controller initialization flow . . . 2094
Figure 645. 10-bit address read access with HEAD10R = 0 . . . 2094

Figure 646. 10-bit address read access with HEAD10R = 1 . . . 2095
Figure 647. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 2096
Figure 648. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 2097
Figure 649. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 2098
Figure 650. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 2100
Figure 651. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 2101
Figure 652. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 2102
Figure 653. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 2106
Figure 654. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 2109
Figure 655. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 2110
Figure 656. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 2111
Figure 657. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . 2112
Figure 658. Bus transfer diagrams for SMBus controller transmitter . . . 2113
Figure 659. Bus transfer diagrams for SMBus controller receiver . . . 2115
Figure 660. I3C block diagram . . . 2139
Figure 661. I3C (primary) controller state and programming sequence diagram . . . 2143
Figure 662. I3C target state and programing sequence diagram . . . 2148
Figure 663. I3C CCC messages, as controller . . . 2162
Figure 664. I3C broadcast ENTDAA CCC, as controller . . . 2163
Figure 665. I3C broadcast, direct read and direct write RSTACT CCC, as controller . . . 2164
Figure 666. I3C CCC messages, as target . . . 2166
Figure 667. I3C broadcast ENTDAA CCC, as target . . . 2167
Figure 668. I3C broadcast DEFTGTS CCC, as target . . . 2168
Figure 669. I3C broadcast DEFGRPA CCC, as target . . . 2169
Figure 670. I3C private read/write messages, as controller . . . 2171
Figure 671. I3C private read/write messages, as target . . . 2172
Figure 672. Legacy I2C read/write messages, as controller . . . 2173
Figure 673. IBI transfer, as controller/target . . . 2174
Figure 674. Hot-join request transfer, as controller/target . . . 2175
Figure 675. Controller-role request transfer, as controller/target . . . 2176
Figure 676. C-FIFO management, as controller . . . 2177
Figure 677. TX-FIFO management, as controller . . . 2179
Figure 678. RX-FIFO management, as controller . . . 2181
Figure 679. S-FIFO management, as controller . . . 2184
Figure 680. RX-FIFO management, as target on the I3C bus . . . 2185
Figure 681. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus . . . 2187
Figure 682. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target . . . 2189
Figure 683. USART block diagram . . . 2247
Figure 684. Word length programming . . . 2251
Figure 685. Configurable stop bits . . . 2253
Figure 686. TC/TXE behavior when transmitting . . . 2255
Figure 687. Start bit detection when oversampling by 16 or 8 . . . 2256
Figure 688. usart_ker_ck clock divider block diagram . . . 2259
Figure 689. Data sampling when oversampling by 16 . . . 2260
Figure 690. Data sampling when oversampling by 8 . . . 2261
Figure 691. Mute mode using Idle line detection . . . 2268
Figure 692. Mute mode using address mark detection . . . 2269
Figure 693. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 2272
Figure 694. Break detection in LIN mode vs. Framing error detection . . . 2273

Figure 695. USART example of synchronous master transmission. 2274
Figure 696. USART data clock timing diagram in synchronous master mode (M bits = 00) 2274
Figure 697. USART data clock timing diagram in synchronous master mode (M bits = 01) 2275
Figure 698. USART data clock timing diagram in synchronous slave mode (M bits = 00) 2276
Figure 699. ISO 7816-3 asynchronous protocol 2278
Figure 700. Parity error detection using the 1.5 stop bits 2280
Figure 701. IrDA SIR ENDEC block diagram. 2284
Figure 702. IrDA data modulation (3/16) - normal mode 2284
Figure 703. Transmission using DMA 2286
Figure 704. Reception using DMA 2287
Figure 705. Hardware flow control between two USARTs 2287
Figure 706. RS232 RTS flow control 2288
Figure 707. RS232 CTS flow control 2289
Figure 708. Wake-up event verified (wake-up event = address match, FIFO disabled) 2292
Figure 709. Wake-up event not verified (wake-up event = address match, FIFO disabled) 2292
Figure 710. LPUART block diagram 2337
Figure 711. LPUART word length programming 2340
Figure 712. Configurable stop bits 2342
Figure 713. TC/TXE behavior when transmitting 2344
Figure 714. lpuart_ker_ck clock divider block diagram 2348
Figure 715. Mute mode using Idle line detection 2352
Figure 716. Mute mode using address mark detection 2353
Figure 717. Transmission using DMA 2355
Figure 718. Reception using DMA 2356
Figure 719. Hardware flow control between two LPUARTs 2357
Figure 720. RS232 RTS flow control 2357
Figure 721. RS232 CTS flow control 2358
Figure 722. Wake-up event verified (wake-up event = address match, FIFO disabled) 2361
Figure 723. Wake-up event not verified (wake-up event = address match, FIFO disabled) 2361
Figure 724. SPI/I2S block diagram 2393
Figure 725. Full-duplex single master/ single slave application 2396
Figure 726. Half-duplex single master/ single slave application 2397
Figure 727. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) 2398
Figure 728. Master and three independent slaves connected in star topology 2399
Figure 729. Master and three slaves connected in circular (daisy chain) topology 2401
Figure 730. Multimaster application 2402
Figure 731. Scheme of NSS control logic 2404
Figure 732. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) 2404
Figure 733. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) 2405
Figure 734. Data clock timing diagram 2407
Figure 735. Data alignment when data size is not equal to 8, 16 or 32 bits 2408
Figure 736. TI mode transfer 2417
Figure 737. Optional configurations of the slave behavior when an underrun condition is detected . 2419
Figure 738. Waveform examples 2427
Figure 739. Master I2S Philips protocol waveforms (16/32-bit full accuracy) 2428

Figure 740. I2S Philips standard waveforms . . . 2428
Figure 741. Master MSB-justified 16- or 32-bit full-accuracy length . . . 2429
Figure 742. Master MSB-justified 16- or 24-bit data length . . . 2429
Figure 743. Slave MSB-justified 16-, 24- or 32-bit data length . . . 2430
Figure 744. LSB-justified 16 or 24-bit data length . . . 2430
Figure 745. Master PCM when the frame length is equal the data length . . . 2431
Figure 746. Master PCM standard waveforms (16 or 24-bit data length) . . . 2431
Figure 747. Slave PCM waveforms . . . 2432
Figure 748. Startup sequence, I2S Philips standard, master . . . 2435
Figure 749. Startup sequence, I2S Philips standard, slave . . . 2436
Figure 750. Stop sequence, I2S Philips standard, master . . . 2436
Figure 751. I^2S clock generator architecture . . . 2437
Figure 752. Data Format . . . 2439
Figure 753. Handling of underrun situation . . . 2441
Figure 754. Handling of overrun situation . . . 2442
Figure 755. Frame error detection, with FIXCH = 0 . . . 2443
Figure 756. Frame error detection, with FIXCH = 1 . . . 2443
Figure 757. SAI functional block diagram . . . 2469
Figure 758. Audio frame . . . 2473
Figure 759. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . 2475
Figure 760. FS role is start of frame (FSDEF = 0) . . . 2476
Figure 761. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . 2477
Figure 762. First bit offset . . . 2477
Figure 763. Audio block clock generator overview . . . 2479
Figure 764. PDM typical connection and timing . . . 2483
Figure 765. Detailed PDM interface block diagram . . . 2484
Figure 766. Start-up sequence . . . 2485
Figure 767. SAI_ADR format in TDM mode, 32-bit slot width . . . 2486
Figure 768. SAI_ADR format in TDM mode, 16-bit slot width . . . 2487
Figure 769. SAI_ADR format in TDM mode, 8-bit slot width . . . 2488
Figure 770. AC’97 audio frame . . . 2491
Figure 771. Example of typical AC’97 configuration on devices featuring at least two embedded SAIs (three external AC’97 decoders) . . . 2492
Figure 772. SPDIF format . . . 2493
Figure 773. SAI_xDR register ordering . . . 2494
Figure 774. Data companding hardware in an audio block in the SAI . . . 2497
Figure 775. Tristate strategy on SD output line on an inactive slot . . . 2499
Figure 776. Tristate on output data line in a protocol like I2S . . . 2500
Figure 777. Overrun detection error . . . 2501
Figure 778. FIFO underrun event . . . 2501
Figure 779. CAN subsystem . . . 2536
Figure 780. FDCAN block diagram . . . 2538
Figure 781. Bit timing . . . 2540
Figure 782. Transceiver delay measurement . . . 2545
Figure 783. Pin control in bus monitoring mode . . . 2546
Figure 784. Pin control in loop-back mode . . . 2549
Figure 785. CAN error state diagram . . . 2550
Figure 786. Message RAM configuration . . . 2551
Figure 787. Standard message ID filter path . . . 2554
Figure 788. Extended message ID filter path . . . 2555
Figure 789. USB peripheral block diagram . . . 2602
Figure 790. Packet buffer areas with examples of buffer description table locations . . . 2608

Figure 791. UCPD block diagram . . . 2650
Figure 792. Clock division and timing elements . . . 2652
Figure 793. K-code transmission . . . 2654
Figure 794. Transmit order for various sizes of data . . . 2655
Figure 795. Packet format . . . 2656
Figure 796. Line format of Hard Reset . . . 2656
Figure 797. Line format of Cable Reset . . . 2657
Figure 798. BIST test data frame . . . 2658
Figure 799. BIST Carrier Mode 2 frame . . . 2658
Figure 800. UCPD BMC transmitter architecture . . . 2659
Figure 801. UCPD BMC receiver architecture . . . 2660
Figure 802. Ethernet high-level block diagram . . . 2693
Figure 803. DMA transmission flow (standard mode) . . . 2696
Figure 804. DMA transmission flow (OSP mode) . . . 2698
Figure 805. Receive DMA flow . . . 2700
Figure 806. Overview of MAC transmission flow . . . 2704
Figure 807. MAC reception flow . . . 2706
Figure 808. Packet filtering sequence . . . 2710
Figure 809. Networked time synchronization . . . 2719
Figure 810. Propagation delay calculation in clocks supporting peer-to-peer path correction . . . 2720
Figure 811. System time update using fine correction method . . . 2730
Figure 812. TCP segmentation offload overview . . . 2747
Figure 813. TCP segmentation offload flow . . . 2748
Figure 814. Header and payload fields of segmented packets . . . 2751
Figure 815. Supported PHY interfaces . . . 2761
Figure 816. SMA Interface block . . . 2761
Figure 817. MDIO packet structure (Clause 45) . . . 2762
Figure 818. MDIO packet structure (Clause 22) . . . 2763
Figure 819. SMA write operation flow . . . 2765
Figure 820. Write data packet . . . 2766
Figure 821. Read data packet . . . 2766
Figure 822. Media independent interface (MII) signals . . . 2768
Figure 823. RMII block diagram . . . 2770
Figure 824. Transmission bit order . . . 2771
Figure 825. Receive bit order . . . 2772
Figure 826. LPI transitions (Transmit, 100 Mbds) . . . 2780
Figure 827. LPI Tx clock gating (when LPITCSE = 1) . . . 2781
Figure 828. LPI transitions (receive, 100 Mbps) . . . 2782
Figure 829. Descriptor ring structure . . . 2803
Figure 830. DMA descriptor ring . . . 2805
Figure 831. Descriptor tail pointer example 1 . . . 2805
Figure 832. Descriptor tail pointer example 2 . . . 2806
Figure 833. Transmit descriptor (read format) . . . 2807
Figure 834. Transmit descriptor write-back format . . . 2812
Figure 835. Transmit context descriptor format . . . 2816
Figure 836. Receive normal descriptor (read format) . . . 2820
Figure 837. Receive normal descriptor (write-back format) . . . 2822
Figure 838. Receive context descriptor . . . 2830
Figure 839. Generation of ETH_DMAISR flags . . . 2846
Figure 840. HDMI-CEC block diagram . . . 2966
Figure 841. Message structure . . . 2966

Figure 842. Blocks . . . 2967
Figure 843. Bit timings . . . 2967
Figure 844. Signal free time . . . 2968
Figure 845. Arbitration phase . . . 2968
Figure 846. SFT of three nominal bit periods . . . 2968
Figure 847. Error bit timing . . . 2969
Figure 848. Error handling . . . 2971
Figure 849. TXERR detection . . . 2972
Figure 850. Block diagram of debug support infrastructure . . . 2983
Figure 851. Product life cycle states and debug authentication . . . 2987
Figure 852. JTAG TAP state machine . . . 2991
Figure 853. CoreSight topology . . . 3012
Figure 854. Trace port interface unit (TPIU) . . . 3091
Figure 855. Embedded cross trigger . . . 3103

1 Documentation conventions

1.1 General information

The STM32H5 devices have an Arm®(a) Cortex®-M33 core.

1.2 List of abbreviations for registers

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

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

1.3 Register reset value

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

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

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

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

1.4 Glossary

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

- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **AHB**: advanced high-performance bus.

2 Memory and bus architecture

2.1 System architecture

The architecture relies on a Arm Cortex-M33 core optimized for execution thanks to an instruction cache having a direct access to the embedded flash memory.

This architecture also features a 32-bit multilayer AHB bus matrix with the interconnections detailed in *Table 1* and *Table 2*.

Table 1. Implementation of masters

Master	Comments	STM32H523/533xx	STM32H562/563/573xx
Cortex-M33 Fast C-bus	Connecting Cortex-M33 (with Arm TrustZone® mainline and FPU) to the internal SRAMs and flash memory through ICACHE	X	X
Cortex-M33 Slow C-bus	Connecting Cortex-M33 (with Arm TrustZone mainline and FPU) to the external memories through ICACHE	X	X
Cortex-M33 S-bus	Connecting the Cortex-M33 (with Arm TrustZone mainline and FPU) to internal SRAMs without latency	3 masters	
	Connecting Cortex-M33 (with Arm TrustZone mainline and FPU) to the external memories through DCACHE	X	X
GPDMA1	-	2 masters	
GPDMA2	-	2 masters	
SDMMC1	-	X	X
SDMMC2	-	-	X
Ethernet MAC	-	-	X(1)

1. Not available on STM32H562xx devices.

Table 2. Implementation of slaves

Slave	Comments	STM32H523/533xx	STM32H562/563/573xx
Flash memory	-	X	X
SRAM1	-	X	X
SRAM2	-	X	
SRAM3	-	X	X
AHB1	Peripherals and BKPSRAM including AHB to APB bridge, and APB peripherals (connected to APB1 and APB2)	X	X
AHB2	Peripherals	X	X
FMC	-	X(1)	X
OCTOSPI1	-	X	X

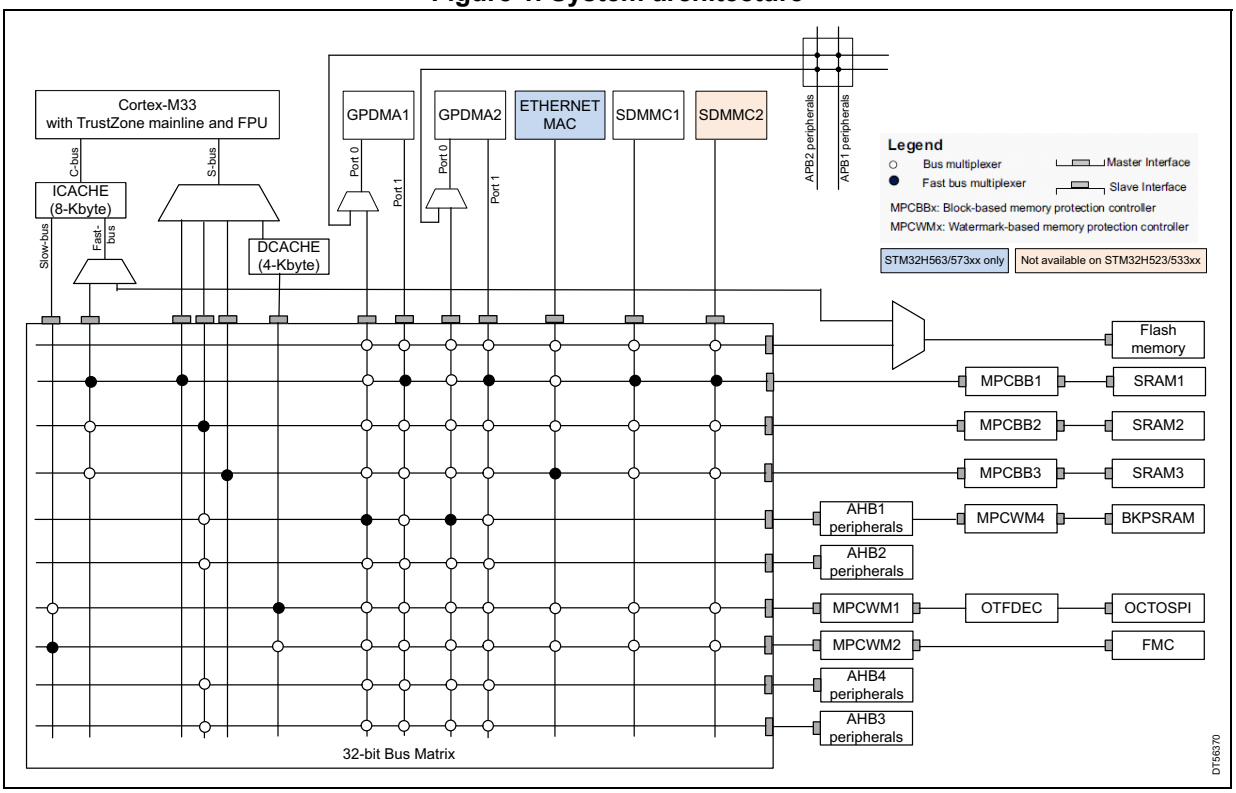
Table 2. Implementation of slaves (continued)

Slave	Comments	STM32H523/533xx	STM32H562/563/573xx
AHB3	Including AHB to APB bridge, and APB peripherals (connected to APB3)	X	X
AHB4	Peripherals	X	X

1. STM32H523/533xx devices do not support SDRAM.

The bus matrix provides access from a master to a slave, enabling concurrent accesses and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in *Figure 1*.

Figure 1. System architecture



2.1.1 Fast C-bus

This bus connects the C-bus of the Cortex-M33 core to the internal flash memory and to the BusMatrix via the instruction cache. This bus is used for instruction fetch and data access to the internal memories mapped in code region. This bus targets the internal flash memory and the internal SRAMs (SRAM1, SRAM2 and SRAM3).

SRAM1, SRAM2 and SRAM3 are accessible on this bus with a continuous mapping.

2.1.2 Slow C-bus

This bus connects the C-bus of the Cortex-M33 core to the BusMatrix via the instruction cache. This bus is used for instruction fetch and data access to the external memories mapped in code region. This bus targets the external memories (FMC and OCTOSPI).

2.1.3 S-bus

This bus connects the system bus of the Cortex-M33 core to the BusMatrix. This bus is used by the core to access data located in a peripheral or SRAM area. This bus targets the internal SRAMs (SRAM1, SRAM2, SRAM3, and BKPSRAM), the AHB1 peripherals including the APB1, APB2, AHB2, AHB3 and AHB4 peripherals.

SRAM1, SRAM2 and SRAM3 are accessible on this bus with a continuous mapping.

Note: The Bus Matrix has a zero latency when accessing SRAM1, SRAM2 and SRAM3.

2.1.4 DCache S-bus

This bus connects the system bus of the Cortex-M33 core to the BusMatrix via the data cache. This bus is used for instruction fetch and data access to the external memories mapped in data region. This bus targets the external memories (FMC and OCTOSPI).

Note: Fetching instructions through this bus is less efficient than fetching instructions through the slow C-bus.

2.1.5 GPDMA1 and GPDMA2 buses

These buses connect the four AHB master interfaces of the GPDMA1 and GPDMA2 to the BusMatrix. These buses target the internal flash memory, the internal SRAMs (SRAM1, SRAM2, SRAM3, and BKPSRAM), the AHB1 peripherals including the APB1 and APB2 peripherals, the AHB2 peripherals, the AHB3 peripherals, AHB4 peripherals and the external memories through FMC or OCTOSPI.

2.1.6 SDMMC1 and SDMMC2[(a)] controllers DMA buses

These buses connect the SDMMC1 and SDMMC2 DMA master interfaces to the BusMatrix. They are used only by the SDMMC1 and SDMMC2 DMA to load/store data from/to the memory. These buses target the data memories: internal flash memory, internal SRAMs (SRAM1, SRAM2 and SRAM3), and external memories through FMC or OCTOSPI.

2.1.7 BusMatrix

The BusMatrix manages the access arbitration between masters. The arbitration uses a round-robin algorithm. This BusMatrix features a fast bus multiplexer used to connect each master to a given slave without latency (see *Figure 1*). For the same master, other slaves undergo a latency of at least one cycle at each new access.

a. SDMMC2 is not available on STM32H523/533xx devices.

2.1.8 AHB/APB bridges

The three AHB/APB bridges provide full synchronous connections between the AHB and the APB buses, allowing flexible selection of the peripheral frequency.

Refer to *Section 2.3.2: Memory map and register boundary addresses* for the address mapping of the peripherals connected to these bridges.

After each device reset, all peripheral clocks are disabled (except for the internal SRAMs and flash memory interfaces). Before using a peripheral, its clock must be enabled in the RCC_AHBxENR and RCC_APBxENR registers.

Note: When a 8- or 16-bit access is performed on an APB register, the access is transformed into a 32-bit access: the bridge duplicates the 8- or 16-bit data to feed the 32-bit vector.

2.1.9 Ethernet MAC[a]

The Ethernet MAC uses a 32-bit bus, connected to the AHB bus matrix. Through the system bus matrices, it can access the internal flash memory, the internal memories, and the external memories through the OCTOSPI and the FMC.

2.2 TrustZone security architecture

The security architecture is based on Arm TrustZone with the Armv8-M mainline extension.

The TrustZone security is activated by the TZEN option bit in the FLASH_OPTR register.

When the TrustZone is enabled, the SAU (security-attribution unit) and IDAU (implementation-defined-attribution unit) defines the access permissions based on secure and non-secure states.

- SAU: Up to eight SAU configurable regions are available for security attribution.
- IDAU: provides a first memory partition as non-secure or non-secure callable attributes. The IDAU memory map partition is not configurable, it is fixed by hardware implementation (refer to *Figure 2: Memory map based on IDAU mapping (STM32H562/563/573xx devices)* and *Figure 3: Memory map based on IDAU mapping (STM32H523/533xx devices)*). It is then combined with the results from the SAU security attribution, and the higher security state is selected.

Based on IDAU security attribution, the flash memory, system SRAMs and peripherals memory spaces are aliased twice for secure and non-secure states. However, the external memories space is not aliased.

2.2.1 Default TrustZone security state

When the TrustZone security is activated by the TZEN option bit in the FLASH_OPTR, the default system security state is detailed below:

- CPU:
  - Cortex-M33 is in secure state after reset. The boot address must be at a secure address.
- Memory map:

a. Not available on STM32H523/533xx and STM32H562xx devices.

  - SAU is fully secure after reset. Consequently, all memory map is fully secure. Up to eight SAU configurable regions are available for security attribution.
- Flash memory:
  - The flash memory security area is defined by watermark user options.
  - Block-based security attributions are non-secure after reset.
- SRAMs:
  - All SRAMs are secure after reset. MPCBBx (block-based memory protection controller) are secure.
- External memories:
  - FMC and OCTOSPI banks are secure after reset. MPCWMx (watermark-based memory protection controller) are secure.
- Peripherals (see *Table 3 and Table 4 for a list of securable and TrustZone-aware peripherals)*
  - Securable peripherals are non-secure after reset.
  - TrustZone-aware peripherals are non-secure after reset. Their secure configuration registers are secure.
- All GPIO are secure after reset.
- Interrupts:
  - NVIC: All interrupts are secure after reset. NVIC is banked for secure and non-secure state.
  - TZIC: All illegal access interrupts are disabled after reset (see *GTZC TrustZone system architecture*).

2.2.2 TrustZone peripheral classification

When the TrustZone security is active, a peripheral can be either securable or TrustZone-aware type as follows:

- Securable: peripheral protected by an AHB/APB firewall gate that is controlled from TZSC controller to define security properties
- TrustZone-aware: peripheral connected directly to AHB or APB bus and implementing a specific TrustZone behavior such as a subset of registers being secure.

Refer to *GTZC TrustZone system architecture* for more details.

The following tables list the securable and TrustZone-aware peripherals within the system.

Table 3. Securable peripherals by TZSC

Bus	Peripheral	STM32H523/533xx	STM32H562/563/573xx
AHB4	OCTOSPI	X	X
	FMC	X	X
	SDMMC2	-	X(1)
	SDMMC1	X	X

Table 3. Securable peripherals by TZSC (continued)

Bus	Peripheral	STM32H523/533xx	STM32H562/563/573xx
AHB2	PKA	X	X
	SAES(2)	X	X
	RNG	X	X
	HASH	X	X
	AES(2)	X	X
	DCMI/PSSI	X	X
	ADC1 / ADC2	X	X
	DAC 1	X	X
AHB1	DCACHE registers	X	X
	ICACHE registers	X	X
	ETHERNET	-	X(1)
	RAMCFG	X	X
	FMAC	-	X
	CORDIC	-	X
	CRC	X	X
APB3	VREFBUF	X	X
	LPTIM1	X	X
	LPTIM3	-	X
	LPTIM4	-	X
	LPTIM5	-	X
	LPTIM6	-	X
	I2C4	-	X
	I2C3	X	X
	LPUART1	X	X
	SPI5	-	X

Table 3. Securable peripherals by TZSC (continued)

Bus	Peripheral	STM32H523/533xx	STM32H562/563/573xx
APB2	USB FS	X	X
	SAI2	-	X
	SAI1	-	X
	SPI6	-	X
	SPI4	X	X
	TIM17	-	X
	TIM16	-	X
	TIM15	X	X
	USART1	X	X
	TIM8	X	X
	SPI1	X	X
	TIM1	X	X

Table 3. Securable peripherals by TZSC (continued)

Bus	Peripheral	STM32H523/533xx	STM32H562/563/573xx
APB1	UCPD	X	X
	FDCAN2	X	X(1)
	FDCAN1	X	X
	LPTIM2	X	X
	DTS	X	X
	UART12	-	X
	UART9	-	X
	UART8	-	X
	UART7	-	X
	HDMI-CEC	X	X
	USART11	-	X
	USART10	-	X
	USART6	X	X
	CRS	X	X
	I3C1	X	X
	I2C2	X	X
	I2C1	X	X
	UART5	X	X
	UART4	X	X
	USART3	X	X
	USART2	X	X
	SPI3/I2S3	X	X
	SPI2/I2S2	X	X
	IWDG	X	X
	WWDG	X	X
	TIM14	-	X
	TIM13	-	X
	TIM12	X	X
	TIM7	X	X
	TIM6	X	X
	TIM5	X	X
	TIM4	X	X
	TIM3	X	X
	TIM2	X	X

1. Not available on STM32H562xx devices.

2. Available only on devices with cryptography (STM32H533/573xx)

Table 4. TrustZone-aware peripherals

Bus	Peripheral	STM32H523/533xx	STM32H562/563/573xx
AHB4	OTFDEC1	X	X
AHB3	EXTI	X	X
	RCC	X	X
	PWR	X	X
AHB2	GPIOI	-	X
	GPIOH	X	X
	GPIOG	X	X
	GPIOF	X	X
	GPIOE	X	X
	GPIOD	X	X
	GPIOC	X	X
	GPIOB	X	X
	GPIOA	X	X
AHB1	GTZC	X	X
	DCACHE	X	X
	ICACHE	X	X
	FLASH	X	X
	GPDMA2	X	X
	GPDMA1	X	X
APB3	TAMP	X	X
	RTC	X	X
	SBS	X	X

2.3 Memory organization

2.3.1 Introduction

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

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

2.3.2 Memory map and register boundary addresses

Figure 2. Memory map based on IDAU mapping (STM32H562/563/573xx devices)

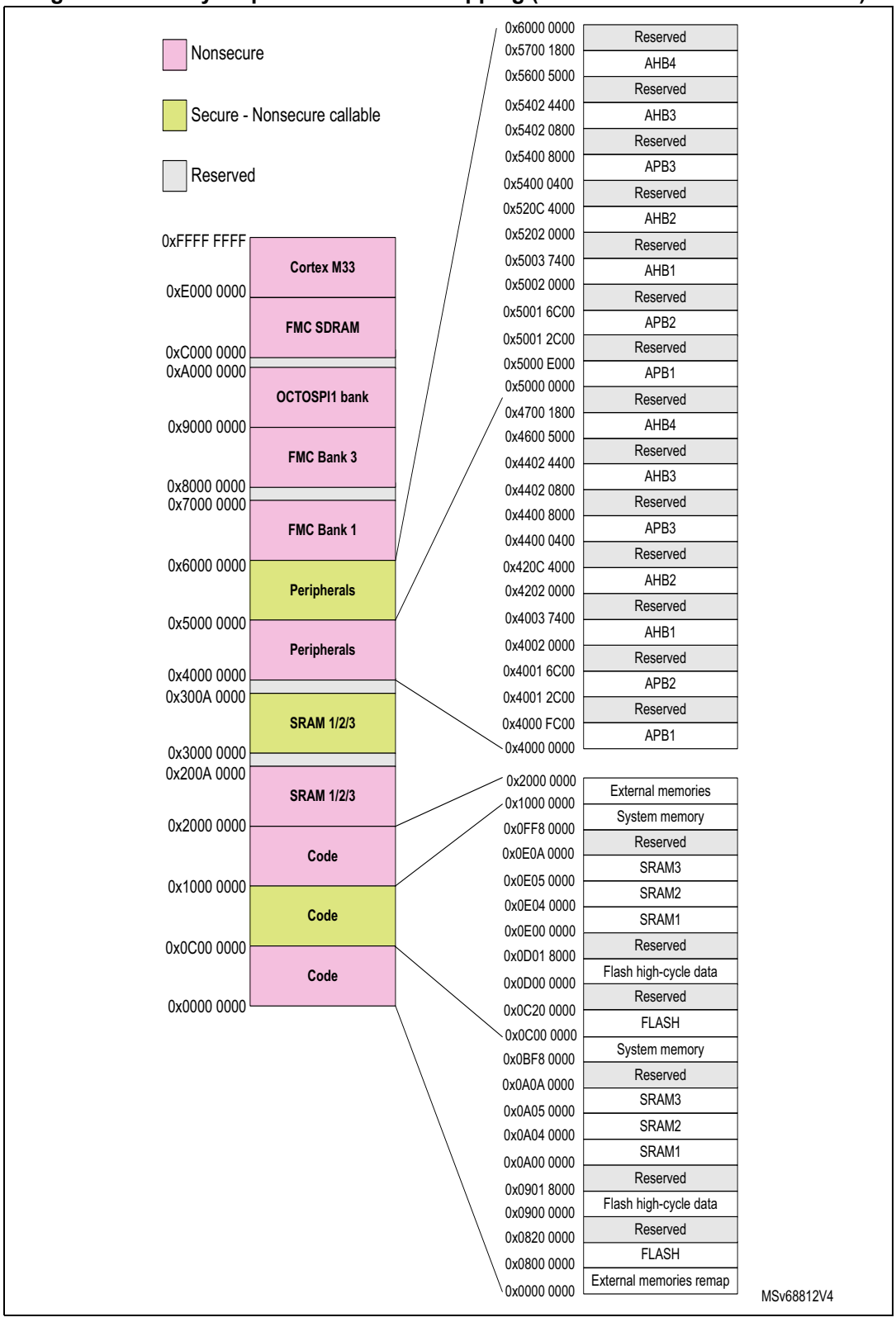
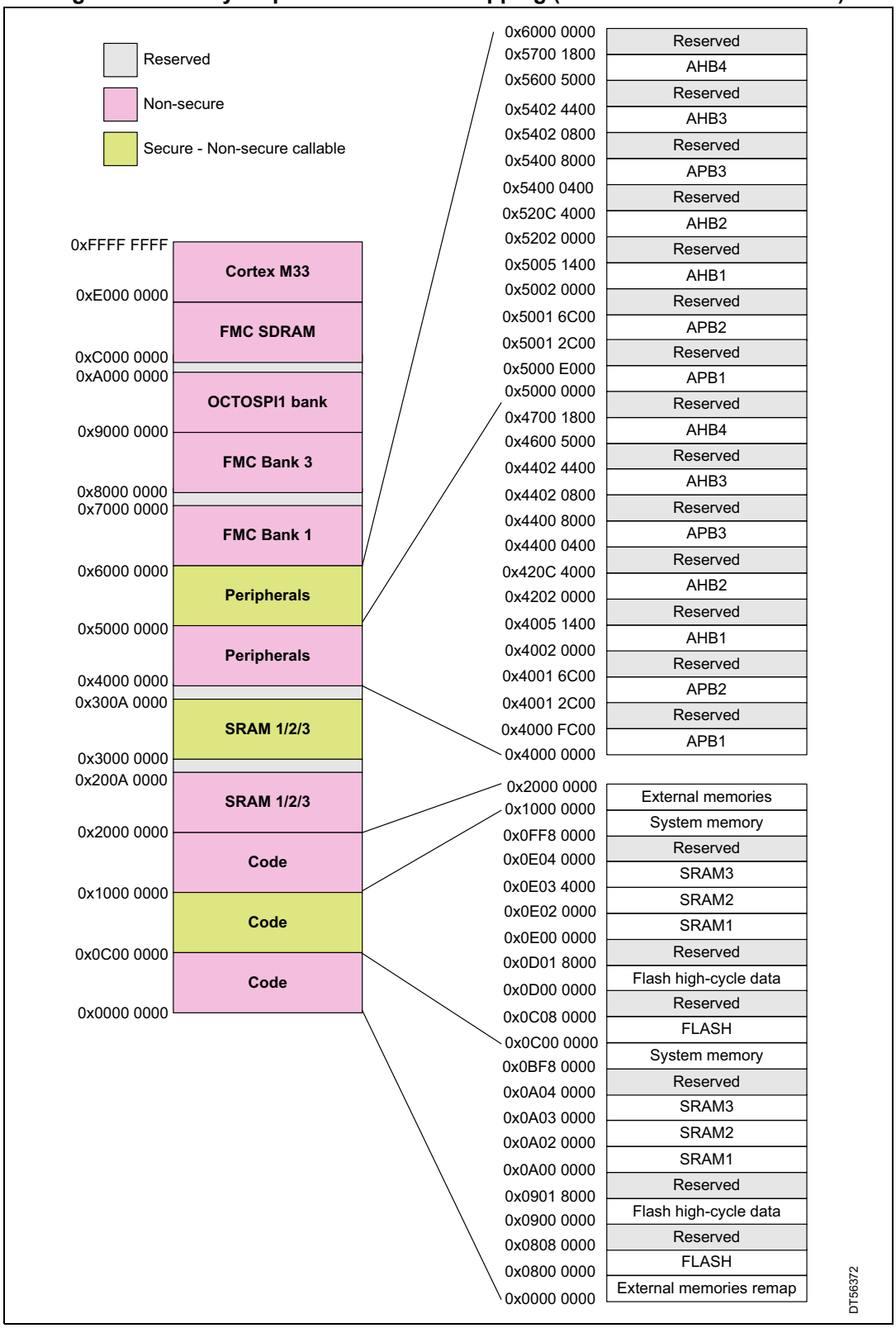


Figure 3. Memory map based on IDAU mapping (STM32H523/533xx devices)



All the memory map areas not allocated to on-chip memories and peripherals are reserved. *Table 5* and *Table 6* give the boundary addresses of the available peripherals.

Table 5. Memory map and peripheral register addresses (STM32H562/563/573xx)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
AHB4	0x5700 1800 - 0x57FF FFFF	0x4700 1800 - 0x47FF FFFF	Reserved	-
	0x5700 1400 - 0x5700 17FF	0x4700 1400 - 0x4700 17FF	OCTOSPI1	OCTOSPI
	0x5700 0800 - 0x5700 13FF	0x4700 0800 - 0x4700 13FF	Reserved	-
	0x5700 0400 - 0x5700 07FF	0x4700 0400 - 0x4700 07FF	FMC	FMC
	0x5600 F800 - 0x5700 03FF	0x4600 F800 - 0x4700 03FF	Reserved	-
	0x5600 F000 - 0x5600 F3FF	0x4600 F000 - 0x4600 F3FF	DLYBOS1	DLYB
	0x5600 9000 - 0x5600 EFFF	0x4600 9000 - 0x4600 EFFF	Reserved	-
	0x5600 8C00 - 0x5600 8FFF	0x4600 8C00 - 0x4600 8FFF	SDMMC2(1)	SDMMC
	0x5600 8800 - 0x5600 8BFF	0x4600 8800 - 0x4600 8BFF	DLYBSD2	DLYB
	0x5600 8400 - 0x5600 87FF	0x4600 8400 - 0x4600 87FF	DLYBSD1	
	0x5600 8000 - 0x5600 83FF	0x4600 8000 - 0x4600 83FF	SDMMC1	SDMMC
	0x5600 5400 - 0x5600 7FFF	0x4600 5400 - 0x4600 7FFF	Reserved	-
	0x5600 5000 - 0x5600 53FF	0x4600 5000 - 0x4600 53FF	OTFDEC1	OTFDEC
	0x5600 0000 - 0x5600 4FFF	0x4600 0000 - 0x4600 4FFF	Reserved	-
AHB3	0x5402 4400 - 0x54FF FFFF	0x4402 4400 - 0x44FF FFFF	Reserved	-
	0x5402 4000 - 0x5402 43FF	0x4402 4000 - 0x4402 43FF	DEBUG	DEBUG
	0x5402 3000 - 0x5402 3FFF	0x4402 3000 - 0x4402 3FFF	Reserved	-
	0x5402 2000 - 0x5402 23FF	0x4402 2000 - 0x4402 23FF	EXTI	EXTI
	0x5402 1000 - 0x5402 1FFF	0x4402 1000 - 0x4402 1FFF	Reserved	-
	0x5402 0C00 - 0x5402 0FFF	0x4402 0C00 - 0x4402 0FFF	RCC	RCC
	0x5402 0800 - 0x5402 0BFF	0x4402 0800 - 0x4402 0BFF	PWR	PWR
	0x5402 0000 - 0x5402 07FF	0x4402 0000 - 0x4402 07FF	Reserved	-
APB3	0x5400 8000 - 0x5400 FFFF	0x4400 8000 - 0x4400 FFFF	Reserved	-
	0x5400 7C00 - 0x5400 7FFF	0x4400 7C00 - 0x4400 7FFF	TAMP	TAMP
	0x5400 7800 - 0x5400 7BFF	0x4400 7800 - 0x4400 7BFF	RTC	RTC
	0x5400 7400 - 0x5400 77FF	0x4400 7400 - 0x4400 77FF	VREFBUF	VREFBUF
	0x5400 5800 - 0x5400 73FF	0x4400 5800 - 0x4400 73FF	Reserved	-

Table 5. Memory map and peripheral register addresses (STM32H562/563/573xx) (continued)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
APB3 (continued)	0x5400 5400 - 0x5400 57FF	0x4400 5400 - 0x4400 57FF	LPTIM6	LPTIM
	0x5400 5000 - 0x5400 53FF	0x4400 5000 - 0x4400 53FF	LPTIM5	
	0x5400 4C00 - 0x5400 4FFF	0x4400 4C00 - 0x4400 4FFF	LPTIM4	
	0x5400 4800 - 0x5400 4BFF	0x4400 4800 - 0x4400 4BFF	LPTIM3	
	0x5400 4400 - 0x5400 47FF	0x4400 4400 - 0x4400 47FF	LPTIM1	
	0x5400 3000 - 0x5400 43FF	0x4400 3000 - 0x4400 43FF	Reserved	-
	0x5400 2C00 - 0x5400 2FFF	0x4400 2C00 - 0x4400 2FFF	I2C4	I2C
	0x5400 2800 - 0x5400 2BFF	0x4400 2800 - 0x4400 2BFF	I2C3	
	0x5400 2400 - 0x5400 27FF	0x4400 2400 - 0x4400 27FF	LPUART1	LPUART
	0x5400 2000 - 0x5400 23FF	0x4400 2000 - 0x4400 23FF	SPI5	SPI
	0x5400 0800 - 0x5400 1FFF	0x4400 0800 - 0x4400 1FFF	Reserved	-
	0x5400 0400 - 0x5400 07FF	0x4400 0400 - 0x4400 07FF	SBS	SBS
	0x5400 0000 - 0x5400 03FF	0x4400 0000 - 0x4400 03FF	Reserved	-
AHB2	0x520C 4000 - 0x53FF FFFF	0x420C 4000 - 0x43FF FFFF	Reserved	-
	0x520C 2000 - 0x520C 3FFF	0x420C 2000 - 0x420C 3FFF	PKA +RAM	PKA
	0x520C 1000 - 0x520C 1FFF	0x420C 1000 - 0x420C 1FFF	Reserved	-
	0x520C 0C00 - 0x520C 0FFF	0x420C 0C00 - 0x420C 0FFF	SAES	SAES
	0x520C 0800 - 0x520C 0BFF	0x420C 0800 - 0x420C 0BFF	RNG	RNG
	0x520C 0400 - 0x520C 07FF	0x420C 0400 - 0x420C 07FF	HASH	HASH
	0x520C 0000 - 0x520C 03FF	0x420C 0000 - 0x420C 03FF	AES	AES
	0x5202 C800 - 0x520B FFFF	0x4202 C800 - 0x420B FFFF	Reserved	-
	0x5202 C400 - 0x5202 C7FF	0x4202 C400 - 0x4202 C7FF	PSSI	PSSI
	0x5202 C000 - 0x5202 C3FF	0x4202 C000 - 0x4202 C3FF	DCMI	DCMI
	0x5202 8800 - 0x5202 BFFF	0x4202 8800 - 0x4202 BFFF	Reserved	-
	0x5202 8400 - 0x5202 87FF	0x4202 8400 - 0x4202 87FF	DAC1	DAC
	0x5202 8000 - 0x5202 83FF	0x4202 8000 - 0x4202 83FF	ADC1 / ADC2	ADC
	0x5202 2400 - 0x5202 7FFF	0x4202 2400 - 0x4202 7FFF	Reserved	-

Table 5. Memory map and peripheral register addresses (STM32H562/563/573xx) (continued)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
AHB2 (continued)	0x5202 2000 - 0x5202 23FF	0x4202 2000 - 0x4202 23FF	GPIOI	GPIO
	0x5202 1C00 - 0x5202 1FFF	0x4202 1C00 - 0x4202 1FFF	GPIOH	
	0x5202 1800 - 0x5202 1BFF	0x4202 1800 - 0x4202 1BFF	GPIOG	
	0x5202 1400 - 0x5202 17FF	0x4202 1400 - 0x4202 17FF	GPIOF	
	0x5202 1000 - 0x5202 13FF	0x4202 1000 - 0x4202 13FF	GPIOE	
	0x5202 0C00 - 0x5202 0FFF	0x4202 0C00 - 0x4202 0FFF	GPIOD	
	0x5202 0800 - 0x5202 0BFF	0x4202 0800 - 0x4202 0BFF	GPIOC	
	0x5202 0400 - 0x5202 07FF	0x4202 0400 - 0x4202 07FF	GPIOB	
	0x5202 0000 - 0x5202 03FF	0x4202 0000 - 0x4202 03FF	GPIOA	
AHB1	0x5003 7400 - 0x51FF FFFF	0x4003 7400 - 0x41FF FFFF	Reserved	-
	0x5003 6400 - 0x5003 73FF	0x4003 6400 - 0x4003 73FF	MPC_WM_BKPRAM	GTZC1 TZSC
	0x5003 3800 - 0x5003 63FF	0x4003 3C00 - 0x4003 63FF	Reserved	-
	0x5003 2400 - 0x5003 37FF	0x4003 2400 - 0x4003 37FF	GTZC1	GTZC1
	0x5003 1800 - 0x5003 23FF	0x4003 1800 - 0x4003 23FF	Reserved	-
	0x5003 1400 - 0x5003 17FF	0x4003 1400 - 0x4003 17FF	DCACHE	DCACHE
	0x5003 0800 - 0x5003 13FF	0x4003 0800 - 0x4003 13FF	Reserved	-
	0x5003 0400 - 0x5003 07FF	0x4003 0400 - 0x4003 07FF	ICACHE	ICACHE
	0x5002 9400 - 0x5003 03FF	0x4002 9400 - 0x4003 03FF	Reserved	-
	0x5002 8000 - 0x5002 93FF	0x4002 8000 - 0x4002 93FF	ETHERNET MAC(1)	ETHERNET
	0x5002 7000 - 0x5002 7FFF	0x4002 7000 - 0x4002 7FFF	Reserved	-
	0x5002 6000 - 0x5002 6FFF	0x4002 6000 - 0x4002 6FFF	RAMCFG	RAMCFG
	0x5002 4000 - 0x5002 5FFF	0x4002 4000 - 0x4002 5FFF	Reserved	-
	0x5002 3C00 - 0x5002 3FFF	0x4002 3C00 - 0x4002 3FFF	FMAC	FMAC
	0x5002 3800 - 0x5002 3BFF	0X4002 3800 - 0x4002 3BFF	CORDIC	CORDIC
	0x5002 3400 - 0x5002 37FF	0x4002 3400 - 0x4002 37FF	Reserved	-
	0x5002 3000 - 0x5002 33FF	0x4002 3000 - 0x4002 33FF	CRC	CRC
	0x5002 2400 - 0x5002 2FFF	0x4002 2400 - 0x4002 2FFF	Reserved	-
	0x5002 2000 - 0x5002 23FF	0x4002 2000 - 0x4002 23FF	FLASH	FLASH
	0x5002 1000 - 0x5002 1FFF	0X4002 1000 - 0x4002 1FFF	DMA2	DMA
	0x5002 0000 - 0x5002 0FFF	0x4002 0000 - 0x4002 0FFF	DMA1	

Table 5. Memory map and peripheral register addresses (STM32H562/563/573xx) (continued)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
APB2	0x5001 6C00 - 0x5001 FFFF	0x4001 6C00 - 0x4001 FFFF	Reserved	-
	0x5001 6400 - 0x5001 6BFF	0x4001 6400 - 0x4001 6BFF	USB_FS RAM	USB
	0x5001 6000 - 0x5001 63FF	0x4001 6000 - 0x4001 63FF	USB_FS	
	0x5001 5C00 - 0x5001 5FFF	0x4001 5C00 - 0x4001 5FFF	Reserved	-
	0x5001 5800 - 0x5001 5BFF	0x4001 5800 - 0x4001 5BFF	SAI2	SAI
	0x5001 5400 - 0x5001 57FF	0x4001 5400 - 0x4001 57FF	SAI1	
	0x5001 5000 - 0x5001 53FF	0x4001 5000 - 0x4001 53FF	SPI6	SPI
	0x5001 4C00 - 0x5001 4FFF	0x4001 4C00 - 0x4001 4FFF	SPI4	
	0x5001 4800 - 0x5001 4BFF	0x4001 4800 - 0x4001 4BFF	TIM17	TIM16 / TIM17
	0x5001 4400 - 0x5001 47FF	0x4001 4400 - 0x4001 47FF	TIM16	
	0x5001 4000 - 0x5001 43FF	0x4001 4000 - 0x4001 43FF	TIM15	TIM15
	0x5001 3C00 - 0x5001 3FFF	0x4001 3C00 - 0x4001 3FFF	Reserved	-
	0x5001 3800 - 0x5001 3BFF	0x4001 3800 - 0x4001 3BFF	USART1	USART
	0x5001 3400 - 0x5001 37FF	0x4001 3400 - 0x4001 37FF	TIM8	TIMx
	0x5001 3000 - 0x5001 33FF	0x4001 3000 - 0x4001 33FF	SPI1 / I2S1	SPI
	0x5001 2C00 - 0x5001 2FFF	0x4001 2C00 - 0x4001 2FFF	TIM1	TIMx
	0x5001 0000 - 0x5001 2BFF	0x4001 0000 - 0x4001 2BFF	Reserved	-
APB1	0x5000 E000 - 0x5000 FFFF	0x4000 E000 - 0x4000 FFFF	Reserved	-
	0x5000 DC00 - 0x5000 DFFF	0x4000 DC00 - 0x4000 DFFF	UCPD1	UCPD
	0x5000 B400 - 0x5000 DBFF	0x4000 B400 - 0x4000 DBFF	Reserved	-
	0x5000 AC00 - 0x5000 B3FF	0x4000 AC00 - 0x4000 B3FF	FDCAN SRAM	FDCAN
	0x5000 A800 - 0x5000 ABFF	0x4000 A800 - 0x4000 ABFF	FDCAN2(1)	
	0x5000 A400 - 0x5000 A7FF	0x4000 A400 - 0x4000 A7FF	FDCAN1(1)	
	0x5000 9800 - 0x5000 A3FF	0x4000 9800 - 0x4000 A3FF	Reserved	-
	0x5000 9400 - 0x5000 97FF	0x4000 9400 - 0x4000 97FF	LPTIM2	LPTIM
	0x5000 9000 - 0x5000 93FF	0x4000 9000 - 0x4000 93FF	Reserved	-
	0x5000 8C00 - 0x5000 8FFF	0x4000 8C00 - 0x4000 8FFF	DTS	DTS
	0x5000 8800 - 0x5000 8BFF	0x4000 8800 - 0x4000 8BFF	Reserved	-
	0x5000 8400 - 0x5000 87FF	0x4000 8400 - 0x4000 87FF	UART12	USART
	0x5000 8000 - 0x5000 83FF	0x4000 8000 - 0x4000 83FF	UART9	
	0x5000 7C00 - 0x5000 7FFF	0x4000 7C00 - 0x4000 7FFF	UART8	
	0x5000 7800 - 0x5000 7BFF	0x4000 7800 - 0x4000 7BFF	UART7	
	0x5000 7400 - 0x5000 77FF	0x4000 7400 - 0x4000 77FF	Reserved	-

Table 5. Memory map and peripheral register addresses (STM32H562/563/573xx) (continued)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
APB1 (continued)	0x5000 7000 - 0x5000 73FF	0x4000 7000 - 0x4000 73FF	HDMI-CEC	HDMI
	0x5000 6C00 - 0x5000 6FFF	0x4000 6C00 - 0x4000 6FFF	USART11	USART
	0x5000 6800 - 0x5000 6BFF	0x4000 6800 - 0x4000 6BFF	USART10	
	0x5000 6400 - 0x5000 67FF	0x4000 6400 - 0x4000 67FF	USART6	
	0x5000 6000 - 0x5000 63FF	0x4000 6000 - 0x4000 63FF	CRS	CRS
	0x5000 5C00 - 0x5000 5FFF	0x4000 5C00 - 0x4000 5FFF	I3C1	I3C
	0x5000 5800 - 0x5000 5BFF	0x4000 5800 - 0x4000 5BFF	I2C2	I2C
	0x5000 5400 - 0x5000 57FF	0x4000 5400 - 0x4000 57FF	I2C1	
	0x5000 5000 - 0x5000 53FF	0x4000 5000 - 0x4000 53FF	UART5	USART
	0x5000 4C00 - 0x5000 4FFF	0x4000 4C00 - 0x4000 4FFF	UART4	
	0x5000 4800 - 0x5000 4BFF	0x4000 4800 - 0x4000 4BFF	USART3	
	0x5000 4400 - 0x5000 47FF	0x4000 4400 - 0x4000 47FF	USART2	
	0x5000 4000 - 0x5000 43FF	0x4000 4000 - 0x4000 43FF	Reserved	-
	0x5000 3C00 - 0x5000 3FFF	0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3	SPI
	0x5000 3800 - 0x5000 3BFF	0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2	
	0x5000 3400 - 0x5000 37FF	0x4000 3400 - 0x4000 37FF	Reserved	-
	0x5000 3000 - 0x5000 33FF	0x4000 3000 - 0x4000 33FF	IWDG	IWDG
	0x5000 2C00 - 0x5000 2FFF	0x4000 2C00 - 0x4000 2FFF	WWDG	WWDG
	0x5000 2400 - 0x5000 2BFF	0x4000 2400 - 0x4000 2BFF	Reserved	-
	0x5000 2000 - 0x5000 23FF	0x4000 2000 - 0x4000 23FF	TIM14	TIMx
	0x5000 1C00 - 0x5000 1FFF	0x4000 1C00 - 0x4000 1FFF	TIM13	
	0x5000 1800 - 0x5000 1BFF	0x4000 1800 - 0x4000 1BFF	TIM12	
	0x5000 1400 - 0x5000 17FF	0x4000 1400 - 0x4000 17FF	TIM7	
	0x5000 1000 - 0x5000 13FF	0x4000 1000 - 0x4000 13FF	TIM6	
	0x5000 0C00 - 0x5000 0FFF	0x4000 0C00 - 0x4000 0FFF	TIM5	
	0x5000 0800 - 0x5000 0BFF	0x4000 0800 - 0x4000 0BFF	TIM4	
	0x5000 0400 - 0x5000 07FF	0x4000 0400 - 0x4000 07FF	TIM3	
	0x5000 0000 - 0x5000 03FF	0x4000 0000 - 0x4000 03FF	TIM2	

1. Not available on STM32H562xx devices.

Table 6. Memory map and peripheral register addresses (STM32H523/533xx)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
AHB4	0x5700 1800 - 0x57FF FFFF	0x4700 1800 - 0x47FF FFFF	Reserved	-
	0x5700 1400 - 0x5700 17FF	0x4700 1400 - 0x4700 17FF	OCTOSPI1	OCTOSPI
	0x5700 0800 - 0x5700 13FF	0x4700 0800 - 0x4700 13FF	Reserved	-
	0x5700 0400 - 0x5700 07FF	0x4700 0400 - 0x4700 07FF	FMC	FMC
	0x5600 F800 - 0x5700 03FF	0x4600 F800 - 0x4700 03FF	Reserved	-
	0x5600 F000 - 0x5600 F3FF	0x4600 F000 - 0x4600 F3FF	DLYBOS1	DLYB
	0x5600 8800 - 0x5600 EFFF	0x4600 8800 - 0x4600 EFFF	Reserved	-
	0x5600 8400 - 0x5600 87FF	0x4600 8400 - 0x4600 87FF	DLYBSD1	DLYB
	0x5600 8000 - 0x5600 83FF	0x4600 8000 - 0x4600 83FF	SDMMC1	SDMMC
	0x5600 5400 - 0x5600 7FFF	0x4600 5400 - 0x4600 7FFF	Reserved	-
	0x5600 5000 - 0x5600 53FF	0x4600 5000 - 0x4600 53FF	OTFDEC1	OTFDEC
	0x5600 0000 - 0x5600 4FFF	0x4600 0000 - 0x4600 4FFF	Reserved	-
AHB3	0x5402 4400 - 0x54FF FFFF	0x4402 4400 - 0x44FF FFFF	Reserved	-
	0x5402 4000 - 0x5402 43FF	0x4402 4000 - 0x4402 43FF	DEBUG	DEBUG
	0x5402 3000 - 0x5402 3FFF	0x4402 3000 - 0x4402 3FFF	Reserved	-
	0x5402 2000 - 0x5402 23FF	0x4402 2000 - 0x4402 23FF	EXTI	EXTI
	0x5402 1000 - 0x5402 1FFF	0x4402 1000 - 0x4402 1FFF	Reserved	-
	0x5402 0C00 - 0x5402 0FFF	0x4402 0C00 - 0x4402 0FFF	RCC	RCC
	0x5402 0800 - 0x5402 0BFF	0x4402 0800 - 0x4402 0BFF	PWR	PWR
	0x5402 0000 - 0x5402 07FF	0x4402 0000 - 0x4402 07FF	Reserved	-
APB3	0x5400 8000 - 0x5400 FFFF	0x4400 8000 - 0x4400 FFFF	Reserved	-
	0x5400 7C00 - 0x5400 7FFF	0x4400 7C00 - 0x4400 7FFF	TAMP	TAMP
	0x5400 7800 - 0x5400 7BFF	0x4400 7800 - 0x4400 7BFF	RTC	RTC
	0x5400 7400 - 0x5400 77FF	0x4400 7400 - 0x4400 77FF	VREFBUF	VREFBUF
	0x5400 4800 - 0x5400 73FF	0x4400 4800 - 0x4400 73FF	Reserved	-
	0x5400 4400 - 0x5400 47FF	0x4400 4400 - 0x4400 47FF	LPTIM1	LPTIM
	0x5400 3400 - 0x5400 43FF	0x4400 3400 - 0x4400 43FF	Reserved	-
	0x5400 3000 - 0x5400 33FF	0x4400 3000 - 0x4400 33FF	I3C2	I3C
	0x5400 2C00 - 0x5400 2FFF	0x4400 2C00 - 0x4400 2FFF	Reserved	-
	0x5400 2800 - 0x5400 2BFF	0x4400 2800 - 0x4400 2BFF	I2C3	I2C
	0x5400 2400 - 0x5400 27FF	0x4400 2400 - 0x4400 27FF	LPUART1	LPUART
	0x5400 0800 - 0x5400 23FF	0x4400 0800 - 0x4400 23FF	Reserved	-
	0x5400 0400 - 0x5400 07FF	0x4400 0400 - 0x4400 07FF	SBS	SBS
	0x5400 0000 - 0x5400 03FF	0x4400 0000 - 0x4400 03FF	Reserved	-

Table 6. Memory map and peripheral register addresses (STM32H523/533xx) (continued)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
AHB2	0x520C 4000 - 0x53FF FFFF	0x420C 4000 - 0x43FF FFFF	Reserved	-
	0x520C 2000 - 0x520C 3FFF	0x420C 2000 - 0x420C 3FFF	PKA + RAM	PKA
	0x520C 1000 - 0x520C 1FFF	0x420C 1000 - 0x420C 1FFF	Reserved	-
	0x520C 0C00 - 0x520C 0FFF	0x420C 0C00 - 0x420C 0FFF	SAES	SAES
	0x520C 0800 - 0x520C 0BFF	0x420C 0800 - 0x420C 0BFF	RNG	RNG
	0x520C 0400 - 0x520C 07FF	0x420C 0400 - 0x420C 07FF	HASH	HASH
	0x520C 0000 - 0x520C 03FF	0x420C 0000 - 0x420C 03FF	AES	AES
	0x5202 C800 - 0x520B FFFF	0x4202 C800 - 0x420B FFFF	Reserved	-
	0x5202 C400 - 0x5202 C7FF	0x4202 C400 - 0x4202 C7FF	PSSI	PSSI
	0x5202 C000 - 0x5202 C3FF	0x4202 C000 - 0x4202 C3FF	DCMI	DCMI
	0x5202 8800 - 0x5202 BFFF	0x4202 8800 - 0x4202 BFFF	Reserved	-
	0x5202 8400 - 0x5202 87FF	0x4202 8400 - 0x4202 87FF	DAC1	DAC
	0x5202 8000 - 0x5202 83FF	0x4202 8000 - 0x4202 83FF	ADC1 / ADC2	ADC
	0x5202 2000 - 0x5202 7FFF	0x4202 2400 - 0x4202 7FFF	Reserved	-
	0x5202 1C00 - 0x5202 1FFF	0x4202 1C00 - 0x4202 1FFF	GPIOH	GPIO
	0x5202 1800 - 0x5202 1BFF	0x4202 1800 - 0x4202 1BFF	GPIOG	
	0x5202 1400 - 0x5202 17FF	0x4202 1400 - 0x4202 17FF	GPIOF	
	0x5202 1000 - 0x5202 13FF	0x4202 1000 - 0x4202 13FF	GPIOE	
	0x5202 0C00 - 0x5202 0FFF	0x4202 0C00 - 0x4202 0FFF	GPIOD	
	0x5202 0800 - 0x5202 0BFF	0x4202 0800 - 0x4202 0BFF	GPIOC	
	0x5202 0400 - 0x5202 07FF	0x4202 0400 - 0x4202 07FF	GPIOB	
	0x5202 0000 - 0x5202 03FF	0x4202 0000 - 0x4202 03FF	GPIOA	

Table 6. Memory map and peripheral register addresses (STM32H523/533xx) (continued)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
AHB1	0x5003 7400 - 0x51FF FFFF	0x4003 7400 - 0x41FF FFFF	Reserved	-
	0x5003 6400 - 0x5003 73FF	0x4003 6400 - 0x4003 73FF	MPC_WM_BKPRAM	GTZC1 TZSC
	0x5003 3800 - 0x5003 63FF	0x4003 3C00 - 0x4003 63FF	Reserved	-
	0x5003 2400 - 0x5003 37FF	0x4003 2400 - 0x4003 37FF	GTZC1	GTZC1
	0x5003 1800 - 0x5003 23FF	0x4003 1800 - 0x4003 23FF	Reserved	-
	0x5003 1400 - 0x5003 17FF	0x4003 1400 - 0x4003 17FF	DCACHE	DCACHE
	0x5003 0800 - 0x5003 13FF	0x4003 0800 - 0x4003 13FF	Reserved	-
	0x5003 0400 - 0x5003 07FF	0x4003 0400 - 0x4003 07FF	ICACHE	ICACHE
	0x5002 7000 - 0x5003 03FF	0x4002 7000 - 0x4003 03FF	Reserved	-
	0x5002 6000 - 0x5002 6FFF	0x4002 6000 - 0x4002 6FFF	RAMCFG	RAMCFG
	0x5002 3400 - 0x5002 5FFF	0x4002 3400 - 0x4002 5FFF	Reserved	-
	0x5002 3000 - 0x5002 33FF	0x4002 3000 - 0x4002 33FF	CRC	CRC
	0x5002 2400 - 0x5002 2FFF	0x4002 2400 - 0x4002 2FFF	Reserved	-
	0x5002 2000 - 0x5002 23FF	0x4002 2000 - 0x4002 23FF	FLASH	FLASH
	0x5002 1000 - 0x5002 1FFF	0X4002 1000 - 0x4002 1FFF	DMA2	DMA
	0x5002 0000 - 0x5002 0FFF	0x4002 0000 - 0x4002 0FFF	DMA1	
APB2	0x5001 6C00 - 0x5001 FFFF	0x4001 6C00 - 0x4001 FFFF	Reserved	-
	0x5001 6400 - 0x5001 6BFF	0x4001 6400 - 0x4001 6BFF	USB_FS RAM	USB
	0x5001 6000 - 0x5001 63FF	0x4001 6000 - 0x4001 63FF	USB_FS	
	0x5001 5000 - 0x5001 5FFF	0x4001 5000 - 0x4001 5FFF	Reserved	-
	0x5001 4C00 - 0x5001 4FFF	0x4001 4C00 - 0x4001 4FFF	SPI4	SPI
	0x5001 4400 - 0x5001 4BFF	0x4001 4400 - 0x4001 4BFF	Reserved	-
	0x5001 4000 - 0x5001 43FF	0x4001 4000 - 0x4001 43FF	TIM15	TIM15
	0x5001 3C00 - 0x5001 3FFF	0x4001 3C00 - 0x4001 3FFF	Reserved	-
	0x5001 3800 - 0x5001 3BFF	0x4001 3800 - 0x4001 3BFF	USART1	USART
	0x5001 3400 - 0x5001 37FF	0x4001 3400 - 0x4001 37FF	TIM8	TIMx
	0x5001 3000 - 0x5001 33FF	0x4001 3000 - 0x4001 33FF	SPI1 / I2S1	SPI
	0x5001 2C00 - 0x5001 2FFF	0x4001 2C00 - 0x4001 2FFF	TIM1	TIMx
	0x5001 0000 - 0x5001 2BFF	0x4001 0000 - 0x4001 2BFF	Reserved	-

Table 6. Memory map and peripheral register addresses (STM32H523/533xx) (continued)

Bus	Secure boundary address	Non-secure boundary address	Peripheral	Register map
APB1	0x5000 E000 - 0x5000 FFFF	0x4000 E000 - 0x4000 FFFF	Reserved	-
	0x5000 DC00 - 0x5000 DFFF	0x4000 DC00 - 0x4000 DFFF	UCPD1	UCPD
	0x5000 B400 - 0x5000 DBFF	0x4000 B400 - 0x4000 DBFF	Reserved	-
	0x5000 AC00 - 0x5000 B3FF	0x4000 AC00 - 0x4000 B3FF	FDCAN SRAM	FDCAN
	0x5000 A800 - 0x5000 ABFF	0x4000 A800 - 0x4000 ABFF	FDCAN2	
	0x5000 A400 - 0x5000 A7FF	0x4000 A400 - 0x4000 A7FF	FDCAN1	
	0x5000 9800 - 0x5000 A3FF	0x4000 9800 - 0x4000 A3FF	Reserved	-
	0x5000 9400 - 0x5000 97FF	0x4000 9400 - 0x4000 97FF	LPTIM2	LPTIM
	0x5000 9000 - 0x5000 93FF	0x4000 9000 - 0x4000 93FF	Reserved	-
	0x5000 8C00 - 0x5000 8FFF	0x4000 8C00 - 0x4000 8FFF	DTS	DTS
	0x5000 7400 - 0x5000 8BFF	0x4000 7400 - 0x4000 8BFF	Reserved	-
	0x5000 7000 - 0x5000 73FF	0x4000 7000 - 0x4000 73FF	HDMI-CEC	HDMI
	0x5000 6800 - 0x5000 6FFF	0x4000 6800 - 0x4000 6FFF	Reserved	-
	0x5000 6400 - 0x5000 67FF	0x4000 6400 - 0x4000 67FF	USART6	USART
	0x5000 6000 - 0x5000 63FF	0x4000 6000 - 0x4000 63FF	CRS	CRS
	0x5000 5C00 - 0x5000 5FFF	0x4000 5C00 - 0x4000 5FFF	I3C1	I3C
	0x5000 5800 - 0x5000 5BFF	0x4000 5800 - 0x4000 5BFF	I2C2	I2C
	0x5000 5400 - 0x5000 57FF	0x4000 5400 - 0x4000 57FF	I2C1	
	0x5000 5000 - 0x5000 53FF	0x4000 5000 - 0x4000 53FF	UART5	USART
	0x5000 4C00 - 0x5000 4FFF	0x4000 4C00 - 0x4000 4FFF	UART4	
	0x5000 4800 - 0x5000 4BFF	0x4000 4800 - 0x4000 4BFF	USART3	
	0x5000 4400 - 0x5000 47FF	0x4000 4400 - 0x4000 47FF	USART2	
	0x5000 4000 - 0x5000 43FF	0x4000 4000 - 0x4000 43FF	Reserved	-
	0x5000 3C00 - 0x5000 3FFF	0x4000 3C00 - 0x4000 3FFF	SPI3 / I2S3	SPI
	0x5000 3800 - 0x5000 3BFF	0x4000 3800 - 0x4000 3BFF	SPI2 / I2S2	
	0x5000 3400 - 0x5000 37FF	0x4000 3400 - 0x4000 37FF	Reserved	-
	0x5000 3000 - 0x5000 33FF	0x4000 3000 - 0x4000 33FF	IWDG	IWDG
	0x5000 2C00 - 0x5000 2FFF	0x4000 2C00 - 0x4000 2FFF	WWDG	WWDG
	0x5000 1C00 - 0x5000 2BFF	0x4000 1C00 - 0x4000 2BFF	Reserved	-
	0x5000 1800 - 0x5000 1BFF	0x4000 1800 - 0x4000 1BFF	TIM12	TIMx
	0x5000 1400 - 0x5000 17FF	0x4000 1400 - 0x4000 17FF	TIM7	
	0x5000 1000 - 0x5000 13FF	0x4000 1000 - 0x4000 13FF	TIM6	
	0x5000 0C00 - 0x5000 0FFF	0x4000 0C00 - 0x4000 0FFF	TIM5	
	0x5000 0800 - 0x5000 0BFF	0x4000 0800 - 0x4000 0BFF	TIM4	
	0x5000 0400 - 0x5000 07FF	0x4000 0400 - 0x4000 07FF	TIM3	
	0x5000 0000 - 0x5000 03FF	0x4000 0000 - 0x4000 03FF	TIM2	

2.3.3 Embedded SRAMs

The STM32H562/563/573xx devices feature up to 644-Kbyte SRAMs:

- 256-Kbyte SRAM1
- 64-Kbyte SRAM2 with ECC
- 320-Kbyte SRAM3 with optional ECC
- 4-Kbyte BKPSRAM with ECC

The STM32H523/533xx devices feature up to 274-Kbyte SRAMs:

- 128-Kbyte SRAM1
- 80-Kbyte SRAM2 with ECC
- 64-Kbyte SRAM3
- 2-Kbyte BKPSRAM with ECC

The SRAMs can be accessed by byte, half-word (16 bits), or full word (32 bits).

The memories can be addressed both by CPU and DMAs. The CPU can access the SRAM1, SRAM2 and SRAM3 through the system bus or through the C-bus, depending on the selected address. The CPU can access the BKPSRAM only through the system bus.

When the TrustZone security is enabled, all SRAMs are secure after reset. The SRAMs can be programmed as non-secure with a block granularity. For more details, refer to *Section 5: Global TrustZone® controller (GTZC)*.

SRAM features are detailed in *Section 6.3.1: Internal SRAMs features*.

2.3.4 Flash memory overview

The flash memory is composed of two distinct physical areas:

- The main memory block, which contains the application program and user data.
- The information block, composed of the following parts:
  - option bytes for hardware and memory protection user configuration
  - system memory, with the ST proprietary code
  - OTP (one-time programmable) area

The flash interface implements instruction and data access based on the AHB protocol. It also implements the logic necessary to carry out the program/erase operations controlled through the FLASH registers plus security access control features. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

2.3.5 Boot modes

At startup, the core jumps to the boot address configured through the BOOT0 pin, the BOOT_ADDR option bytes, the TZEN option bit, debug request and the product state.

If BOOT_ADDR is not yet configured, dedicated libraries (programmed during ST production), can be used for secure boot. They are located in system flash memory:

- ST libraries in system flash memory assist the application software boot with special features such as secure boot and secure firmware install (SFI-RSS).
- ST iROT (immutable root of trust) secure software in user flash memory are used for secure firmware update and provisioning (SFU).

If a debugger is attached to the product, the entry point is a debug authentication policy, used to unlock the device when attached to debugger. Digital signature must be provided to perform a regression to product state, where debug is allowed.

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. Refer to AN2606 “*STM32 microcontroller system memory boot mode”*.

3 System security

The STM32H5 devices are designed with a comprehensive set of security features, some of them based on the standard Arm TrustZone® technology.

These security features simplify the process of evaluating IoT devices against security standards. They also significantly reduce the cost and complexity of software development for OEM and third-party developers, by facilitating the reuse, improving the interoperability, and minimizing the API fragmentation.

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

3.1 Key security features

- Resource isolation using privilege mode and Armv8-M mainline security extension of Cortex-M33, extended to securable I/Os, memories, and peripherals
- Secure firmware installation (SFI) with device unique cryptographic key pair
  - Leveraging the on-chip immutable bootloader that supports the image download through USART, USB, I^2C, I^3C, SPI, FDCAN, and JTAG
- Boot entry: the platform makes it possible to select between native immutable root of trust or proprietary boot entry (in user flash memory).
- Security services (in system flash): the platform comes with native security services, embedded in the system memory to manage the root of trust services. Native root of trust services takes care of: platform security, including secure boot, secure updates of next boot level (uROT: updatable root of trust), secure debug control (debug reopening, regression control). The services can be personalized for each OEM, using the provisioning tools.
- Temporal isolation: boot levels are isolated thanks to HDPL (hide protect level) monotonic counter.
- Secure storage, featuring:
  - Five nonvolatile areas dedicated to secure storage, protected with HDPL and TrustZone.
  - Battery-powered volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in the AES engines
  - Device 96-bit unique ID and JTAG 32-bit device-specific ID
  - Secure storage can rely upon the SAES engine to encrypt the stored data, to the benefit of DHUK properties. All data encrypted with the DHUK benefit of temporal and runtime isolation (HDPL and TrustZone), RHUK, EPOCH (version counter) properties.

  a) Data are isolated: all data encrypted/decrypted relying upon SAES+DHUK benefit of the variation of the DHUK for each different isolated area (HDPL0, HDPL1, HDPL2, HDPL3S, HDPL3NS).

  b) RHUK: hardware secret nonvolatile, unique per device keys (copy protection).

  c) EPOCH: counter incremented each time a regression is done (antirollback).
- General purpose cryptographic acceleration
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes

  - Secure AES 256-bit security coprocessor, supporting ECB, CBC, CTR, GCM, and CCM chaining modes with side-channel countermeasures and mitigations
  - HASH processor, supporting SHA-1 checksums and SHA-2 secure hash (SHA2, SHA2-384, SHA2-512)
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures, and mitigations when manipulating secrets
  - True random number generator (RNG), NIST SP800-90B precertified
- On-the-fly decryption of encrypted image stored on external flash memory connected through the OCTOSPI
  - Almost-zero latency with standard NOR flash memories
  - Can be used to encrypt the image using the device unique secret keys
  - Automatic key-erase in case of tamper
- New flexible life cycle scheme
  - Allows product maintenance in-the-field (debug reopening or regressions)
  - Allows protected firmware distribution in up to three steps: immutable root of trust, secure and nonsecure application
- Active tamper and protection against temperature, voltage, and frequency attacks
  - Up to eight active inputs, eight active output tamper pins, available in different power modes (refer to TAMP low-power modes)

3.2 Secure install

The secure firmware install (SFI) is an immutable secure service embedded by STMicroelectronics in the devices. The SFI allows secure and counted installation of OEM firmware in an untrusted production environment (such as OEM contract manufacturer).

The confidentiality of the installed images written in the internal flash memory or encrypted in an external flash memory, is also protected, using the AES.

The SFI native service leverages the following hardware security features:

- secure boot (see *Section 3.3*)
- resource isolation using TrustZone (see *Section 3.6*)
- temporal isolation using hide protection (see *Section 3.7.1*)
- secure execution (see *Section 3.8*)
- secure storage, with associated cryptographic engines (see *Section 3.9* and *Section 3.10*) and flash memory dedicated areas

Further information can be found in AN4992 “*Overview secure firmware install (SFI)”.*

3.3 Secure boot

Secure boot is an immutable code that is always executed after a system reset. As a root of trust, this code checks the device static protections and activates available device runtime protections, reducing the risk of invalid or malicious code running on the platform. As root of trust, the secure boot also checks the integrity and authenticity of the next level firmware before executing it.

The STM32H533/73xx offer two options (using BOOT_UBE) to execute the immutable code after a reset:

- Security services when ST-iROT (code natively present in the system memory) is selected managing the secure boot of the next boot level. ST-iROT takes care of the next level integrity, authenticity using configuration done by OEMs. The next stage can be an ST updatable service (ST-uROT), or a proprietary one (uROT)
- Proprietary boot entry when the OEM manages the full chain (OEM-iROT), to be installed in user flash memory with proper security activation.

The actual functions of the secure boot depend on the availability of TrustZone features, and on the firmware stored in the device. The secure boot typically initializes the secure storage, and installs on-the-fly decryption keys in the OTFDEC, to be able to use encrypted firmware stored in an external flash memory.

TrustZone application, supported by the STM32 ecosystem, provides a root of trust solution, including secure boot functions.

In the devices, the secure boot benefits of hardware security features:

- resource isolation using TrustZone (see *Section 3.6*)
- temporal isolation using hide protection levels (HDPL) (see *Section 3.7.1*)
- secure execution (see *Section 3.8*)
- secure install and update (see *Section 3.2* and *Section 3.4*)
- per domain secure storage, with associated cryptographic engines if available (see *Section 3.9* and *Section 3.10*)

This section describes the features specifically designed for secure boot.

3.3.1 Unique boot entry

Thanks to the unique boot entry (BOOT_UBE), it is possible to select the boot entry point between security services in system flash (ST-iROT) and proprietary boot entry (OEM-iROT). This selection is possible only for products embedding cryptographic acceleration (STM32H533/73xx), when TrustZone is enabled (TZEN = 0xB4).

When TrustZone is enabled and SECBOOT_LOCK secure option bit is cleared, the application selects a boot entry point located either in the system flash memory (see the next section), or in the secure user flash memory, at the address defined by SECBOOTADD option bytes.

When TrustZone is enabled and SECBOOT_LOCK secure option bit is set, the device unique boot entry is the unmodifiable secure address defined by SECBOOTADD option bytes. The application cannot modify these option bytes when SECBOOT_LOCK is set.

When TrustZone is disabled (TZEN = 0xC3) and NSBOOT_LOCK option bit is cleared, the application selects a boot entry point located in the nonsecure user flash memory, at the address defined by NSBOOTADD option bytes.

When TrustZone is disabled and NSBOOT_LOCK option bit is set, the device unique boot entry is the unmodifiable nonsecure address defined by NSBOOTADD option bytes. The application cannot modify these option bytes when NSBOOT_LOCK is set.

Note: SECBOOT_LOCK and NSBOOT_LOCK can be changed only when PRODUCT_STATE = Open, Provisioning, or Regression.

For more information on the boot mechanisms, refer to *Section 4: Boot modes*.

3.3.2 Immutable root of trust in system flash memory

The immutable root-of-trust code stored in the system flash memory is first used to perform SFI, allowing secure and counted installation of OEM firmware in untrusted production environments (such as OEM contract manufacturer).

The immutable code also includes secure runtime services that can be called at runtime, when a secure application sets the SBS_RSSCMDR register to a non-null value, before triggering a system reset. This runtime feature is deactivated when the BOOT_LOCK secure option bit is set or when the PRODUCT_STATE is different from “Open”.

The STM32H5 products feature ST-DA (debug authentication), to manage the debug authentication control feature, allowing to control the debug reopening and regressions of the product for after sales (field return) of the product.

The products embedding cryptographic acceleration feature ST-iROT (immutable root of trust), to handle the secure provisioning, secure boot, secure updates of the first updatable level of the platform.

3.4 Secure update

The secure firmware update is a secure service that runs after a secure boot. Its actual functions depend on the availability of the TrustZone features, and on the firmware stored in the device.

Cortex-M based application processors is commonly used to run trusted boot and a trusted OS to create a Trusted Execution Environment (TEE). The trusted boot and TEE application, supported by the STM32 ecosystem, allow the update of the microcontroller built-in program with new firmware versions, adding new features and correcting potential issues. The update process is performed in a secure way, to prevent unauthorized updates and access to confidential on-device data.

A firmware update can be done either on a single firmware image including both secure and nonsecure parts, or on the secure (respectively nonsecure) part of the firmware image, independently.

In the devices, the secure update application leverages the same hardware security as the firmware install described in *Section 3.2*.

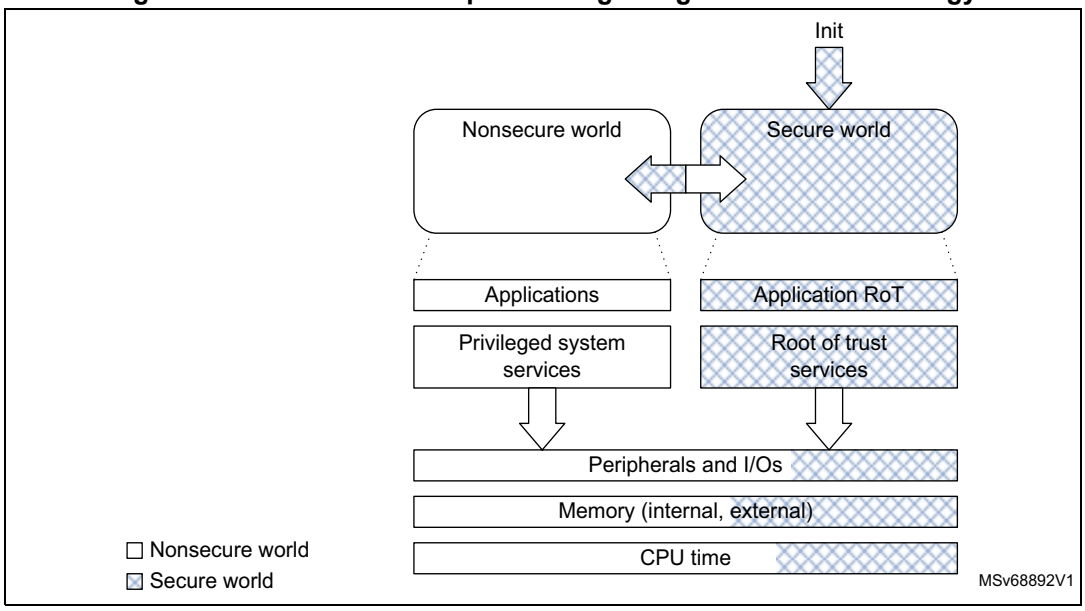
3.5 Resource isolation using hide protection levels

The hardware and software resources used to boot can be isolated. This is called temporal isolation, based on a monotonic counter, taking care of incrementing the levels. When the counter is incremented, the resources (code and data) of the previous levels are hidden.

3.6 Resource isolation using TrustZone

The hardware and software resources can be partitioned so that they exist in the secure or in the nonsecure world, as shown in *Figure 4*.

Figure 4. Secure/nonsecure partitioning using TrustZone technology



Note: The initial partitioning of the platform is under the responsibility of the secure firmware executed after device reset.

Thanks to this resource isolation technology, the secure world can be used to protect critical code against intentional or unintentional tampering from the more exposed code running in the nonsecure world.

Note: The secure code is typically small and rarely modified, while the nonsecure code is more exposed, and prone to firmware updates.

3.6.1 TrustZone security architecture

The Armv8-M TrustZone technology is a comprehensive hardware architecture that proposes to developers a comprehensive, holistic protection across the entire processor and system. The device TrustZone hardware features include:

- The Armv8-M mainline security extension of Cortex-M33, enabling a new processor secure state, with its associated secure interrupts
- The dynamic allocation of memory and peripherals to TrustZone using eight security attribution unit (SAU) regions of Cortex-M33
- A global TrustZone framework (GTZC), extending the TrustZone protection against transactions coming from masters in the system different from Cortex-M33
- TrustZone-aware embedded flash memory and peripherals

Note: The TZEN option bit in the FLASH_OPTSR2_PRG register activates TrustZone security.

3.6.2 Armv8-M security extension of Cortex-M33

The Arm security extension of the Cortex-M33 is an evolution, not a revolution. It uses the programmer model from earlier Cortex-M subfamilies like Cortex-M4. Indeed, Armv8-M is architecturally similar to Armv7-M, using the same 32-bit architecture, the same memory mapped resources protected with an MPU. Armv8-M also uses the nested vectored interrupt controller (NVIC).

The implementation on STM32H5 devices is composed of the following features:

- A new processor state, with almost no additional code/cycle overhead (as opposed to Armv8-A TrustZone) that uses a dedicated exception routine to trigger a secure/nonsecure world change
- Two memory map views of a shared 4-Gbyte address space
- A low interrupt latency for both secure and nonsecure domains, and a new interrupt configuration for security grouping and priority setting
- Separated exception vector tables for the secure and nonsecure exceptions
- Micro-coded context preservation
- Banking of specific registers across secure/nonsecure states, including stack pointers with stack-limit checkers
- Banking of the following Cortex-M33 programmable components (two separate units for secure and nonsecure):
  - SysTick timer
  - MPU configuration registers (12 MPU regions in secure, eight in nonsecure)
  - some of the system control block (SCB) registers
- New system exception (SecureFault) for handling of security violations
- Configurable debug support, as defined in *Section 3.12*

For more information, refer to the Cortex-M33 programming manual (PM0264).

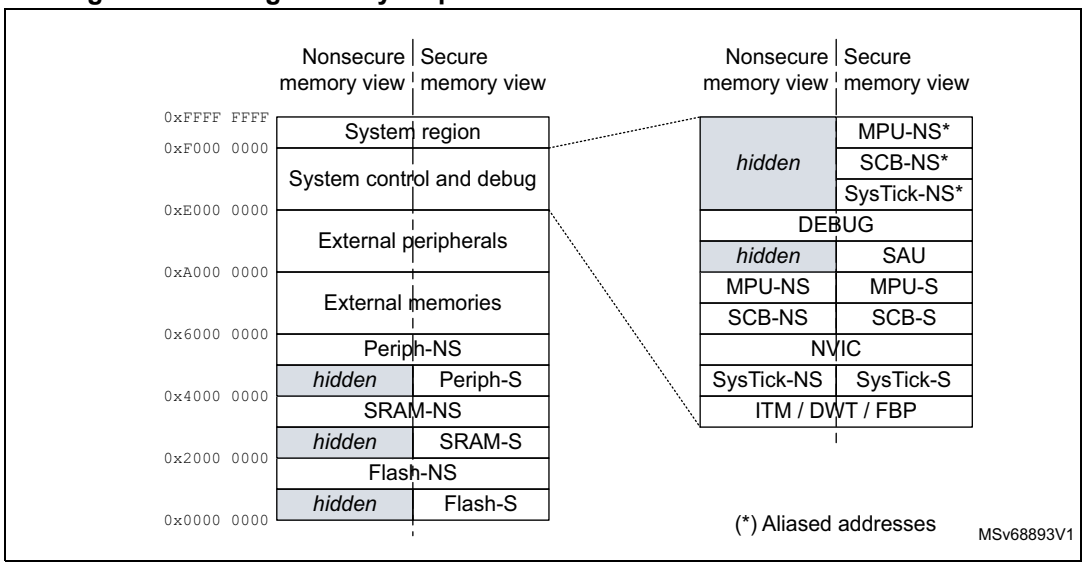
3.6.3 Memory and peripheral allocation using IDAU/SAU

Security attributes

As illustrated in *Figure 5*, the Armv8-M nonsecure memory view is similar to Armv7-M (found in Cortex-M4), with the difference that the secure memory is hidden. The secure memory view shows the flash memory, SRAM and peripherals that are accessible only while the Cortex processor executes in Secure state.

Note: Figure 5 shows the 32-bit address space after SAU configuration, as viewed by the secure code.

Figure 5. Sharing memory map between CPU in secure and nonsecure state

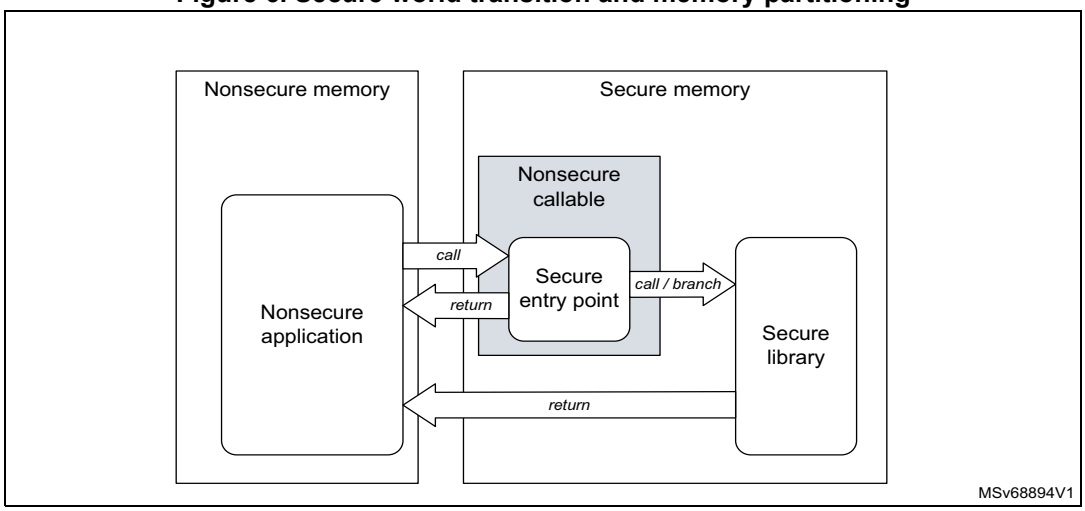


The Cortex processor state (and the associated rights) depends upon the security attribute assigned to the memory region where it is executed:

- A processor in a nonsecure/secure state executes only from nonsecure (NS)/secure (S) program memory.
- When running in secure state, the processor can access data from both S and NS memories. When running in nonsecure state, the CPU is limited to nonsecure memories.

To manage transitions to the secure world, developers must create nonsecure callable (NSC) regions containing valid entry points to the secure libraries. The first instruction in these entry points must be the new secure gate (SG) instruction, used by the nonsecure code to call a secure function (see *Figure 6*).

Figure 6. Secure world transition and memory partitioning



Programming security attributes

In Cortex-M33, the static implementation defined attribution unit (IDAU) works with the programmable security attribution unit (SAU) to assign a specific security attribute (S, NS, or NSC) to a specific address, as shown in *Table 7*.

Table 7. Configuring security attributes with IDAU and SAU

IDAU security attribution	SAU security attribution(1)	Final security attribution
Nonsecure	Secure	Secure
	Secure-NSC	Secure-NSC
	Nonsecure	Nonsecure
Secure-NSC	Secure	Secure
	Nonsecure	Secure-NSC

1. Defined regions are aligned to 32-byte boundaries.

The the Cortex-M33 can configure the SAU only in the secure-privilege state. When the TrustZone is enabled, the SAU defaults all addresses as secure (S). A secure boot application can then program the SAU to create NSC or NS regions, as shown in *Table 7*.

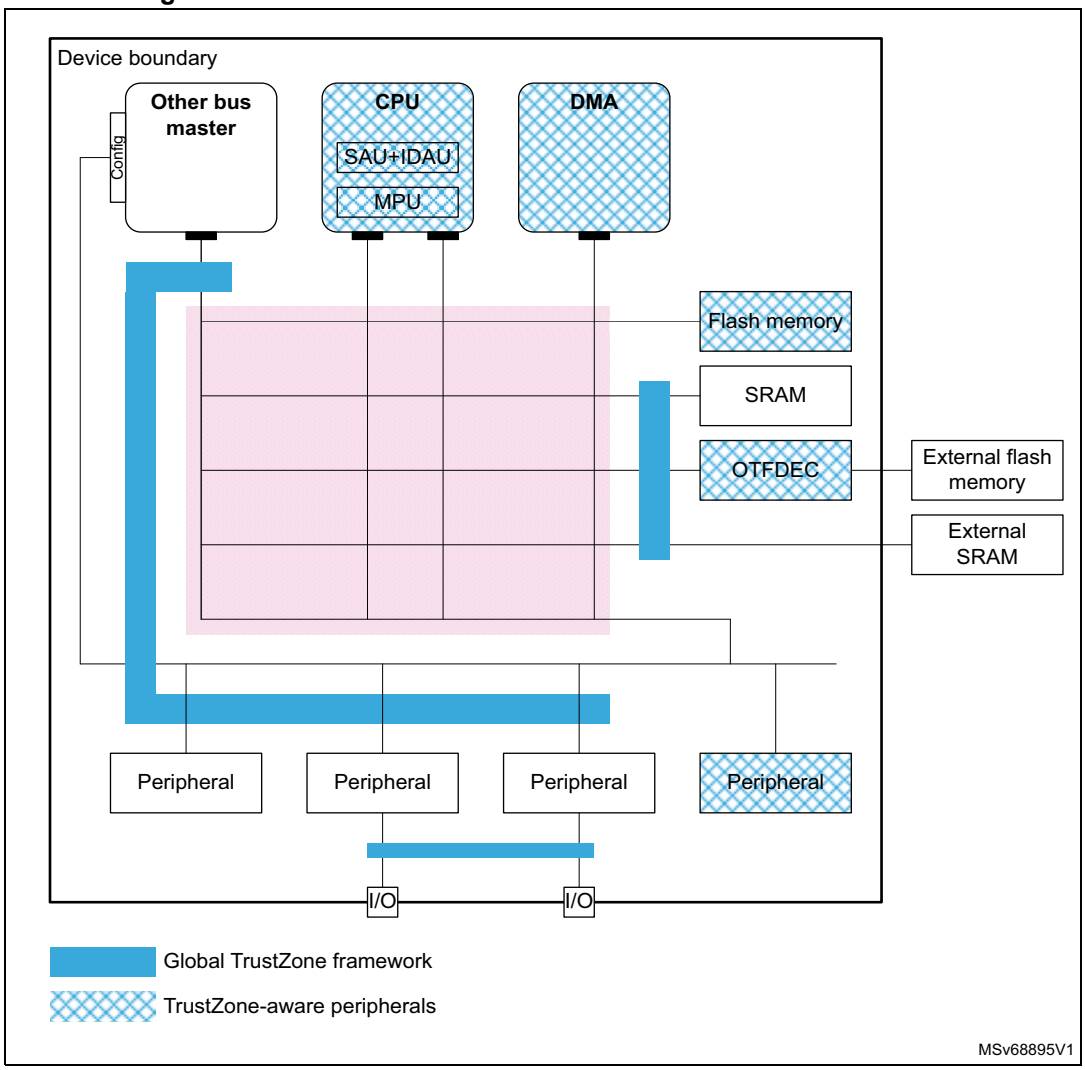
Note: The SAU/IDAU settings are applicable only to the Cortex-M33*. The other masters, like DMA, are not affected by these policies.*

3.6.4 Memory and peripheral allocation using GTZC

Global TrustZone framework architecture

On top of the Armv8-M TrustZone security extension in Cortex-M33, the devices embed complementary security features that reinforce, in a flexible way, the isolation between secure and nonsecure worlds. Unlike the SAU/IDAU, the GTZC can protect legacy memories and peripherals against nonsecure transactions coming from other masters than the Cortex-M33.

Figure 7. Global TrustZone framework and TrustZone awareness



Securing peripherals with TZSC

When the TrustZone security is active, a peripheral is either securable through the GTZC, or is natively TrustZone-aware, as shown in the previous figure:

- A securable peripheral or memory is protected by an AHB/APB firewall gate, controlled by the TrustZone security controller (TZSC).
- A TrustZone-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific TrustZone behavior, such as a subset of secure registers or a secure memory area.

When a securable peripheral is made secure-only with the GTZC, if this peripheral is master on the interconnect, it automatically issues secure transactions. The SDMMC is an example of securable master. TrustZone-aware AHB masters like Cortex-M33 or DMAs, drive a secure signal in the AHB interconnect, according to their security mode, independently to the GTZC.

Note: Like with TrustZone, a peripheral can be made privileged-only with TZSC (see Section 3.7.2). In this case, if this peripheral is master on the interconnect, it automatically issues privileged transactions.

Securing memories with TZSC and MPCBB

The TZSC block in GTZC provides the capability to manage the security and privilege for all securable external memories, programming the MPCWM resources as defined in *Table 8*.

Table 8. MPCWMx resources

Memory	MPC resource	Type of filtering	Number of regions	Default security	On-the-fly decryption(1)
OCTOSPI1	MPCWM1	Nonsecure privileged or unprivileged region (watermarks)	2	Secure privileged(2)	Yes
FMC_NOR bank	MPCWM2		2		No
FMC_NAND / FMC_SDRAM(3) bank	MPCWM3		1		
Backup SRAM (BKPSRAM)/FMC_SDRAM(3)	MPCWM4		1		

1. Using the OTFDEC.
2. Assuming TrustZone is activated on the device, nonsecure unprivileged otherwise.
3. FMC_SDRAM is not available on STM32H523/33xx devices.

The MPCBB resources in GTZC provide the capability to configure the security and privilege of embedded SRAM blocks, as defined in *Table 9* and *Table 10*.

Table 9. MPCBBx resources (STM32H562/63/73xx devices)

Memory	MPC resource	Type of filtering	Memory size	Block size	Number of super-blocks	Default security
SRAM1	GTZC1_MPCBB1	Block based, managing security and privilege	256 KB	512(1) bytes	16	Secure privileged(2)
SRAM2	GTZC1_MPCBB2		64 KB		4	
SRAM3	GTZC1_MPCBB3		320 KB		20	

1. Blocks are grouped in super-blocks of 32 consecutive blocks, to manage the configuration locking.
2. Assuming TrustZone is activated on the device, nonsecure unprivileged otherwise.

Table 10. MPCBBx resources (STM32H523/33xx devices)

Memory	MPC resource	Type of filtering	Memory size	Block size	Number of super-blocks	Default security
SRAM1	GTZC1_MPCBB1	Block based, managing security and privilege	128 KB	512(1) bytes	8	Secure privileged(2)
SRAM2	GTZC1_MPCBB2		80 KB		5	
SRAM3	GTZC1_MPCBB3		64 KB		4	

1. Blocks are grouped in super-blocks of 32 consecutive blocks, to manage the configuration locking.
2. Assuming TrustZone is activated on the device, nonsecure unprivileged otherwise.

Applying GTZC configurations

The TZSC and MPCBB blocks can be used in one of the following ways:

- statically programmed during the secure boot, locked and not changed afterwards
- dynamically reprogrammed using a specific application code or real-time kernel

When the dynamic option is selected and the configuration is not locked:

- MPCBB secure blocks or MPCWM nonsecure region size can be changed by a secure software. This software must be privileged for MPCWM, can be unprivileged if the particular block is not privileged-only.
- The secure (respectively privilege) state of each peripheral can be changed writing to GTZC_TZSC_SECCFRGx (respectively GTZC_TZSC_PRIVCFGRx) registers.

Securing peripherals with TZSC

The TZSC block in GTZC provides the capability to manage the security and the privilege for all securable peripherals. The list of these peripherals can be found in *Section 5: Global TrustZone® controller (GTZC)*.

Note: When the TrustZone is deactivated, the resource isolation hardware GTZC can still be used to isolate peripherals to privileged code only (see Section 3.7.2).

When the TrustZone is activated, peripherals are set as nonsecure and unprivileged after reset.

TrustZone-aware peripherals

The devices include the following TrustZone-aware peripherals:

- GPIOA to GPIOI
- GTZCx_MPCBB, GTZCx_TZIC and GTZCx_TZSC (GTZC blocks)
- OTFDEC1, writable only in secure if TZEN is enabled
- EXTI
- Flash memory
- RCC and PWR
- GPDMA1 and GPDMA2
- SBS registers
- RTC and TAMP
- MCU debug unit DBGMCU

The way illegal accesses to those peripherals are monitored through the TZIC registers is described in *Section 5: Global TrustZone® controller (GTZC)*.

For more details, refer to *Section 3.6.5*.

TrustZone illegal access controller (TZIC)

The TZIC block in GTZC gathers all illegal access events originated from sources either protected by GTZC or TrustZone-aware, generating one global secure interrupt towards the NVIC.

TZIC is available only when the system is TrustZone enabled (TZEN = 0xB4). All accesses to TZIC registers must be secured.

For each illegal event source, a status flag and a clear bit exist. Each illegal event can be masked, not generating an interrupt toward the NVIC.

Note: By default, all events are masked.

3.6.5 Managing security in TrustZone-aware peripherals

This section gives more details on how the security is implemented in the TrustZone-aware peripherals listed in the previous section.

Embedded flash memory

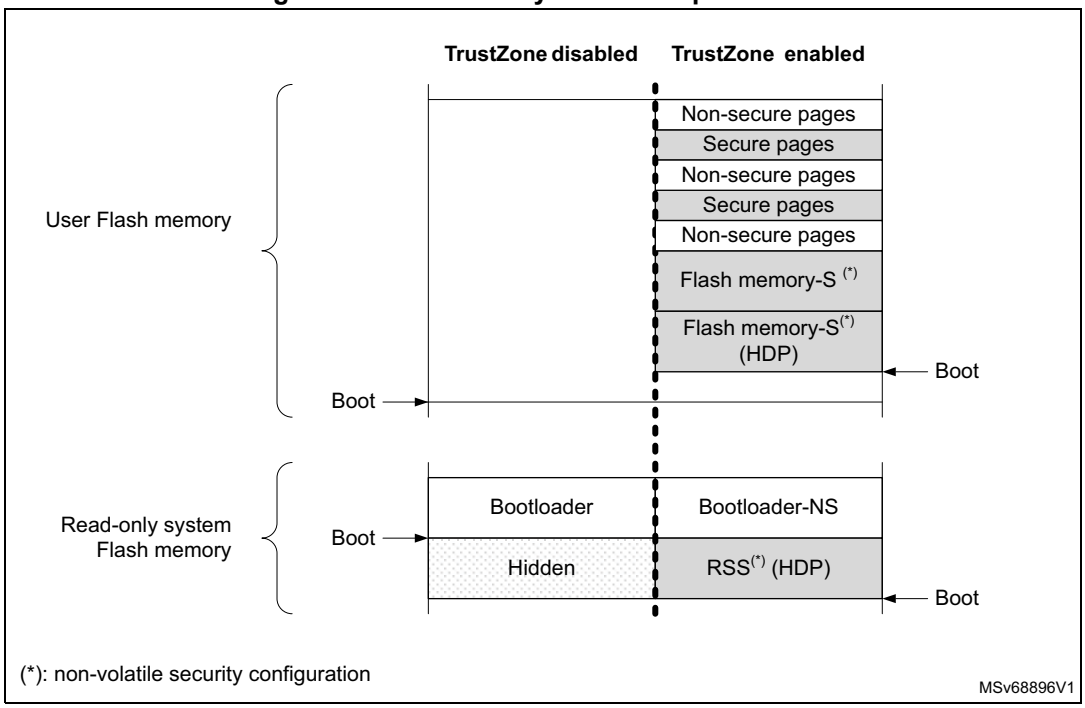
When the TrustZone security is enabled through option bytes (TZEN = 0xB4), the whole memory is secure after reset. The following protections, shown in *Figure 8*, are available to the application:

- nonvolatile user secure areas, defined with nonvolatile secure user option bytes
  - watermark-based secure only area (x2)
  - secure hide protection (HDP) area, stickily hidden after boot (x2)
- volatile user secure pages, defined with volatile secure registers (lost after reset)
  - Any page set as nonsecure (example: outside watermark-based secure only area), can be set as secure on-the-fly using the block-based configuration registers.

Note: All areas are aligned on the page granularity of the flash memory, which can be configured as secure while it is tagged as nonsecure in Cortex-M33 IDAU/SAU. In this case, nonsecure accesses by the CPU to the flash memory are denied.

Erase or program operations can be available to secure/nonsecure code only for secure/nonsecure pages or memory. A flash memory is considered secure if at least one page is secure.

Figure 8. Flash memory TrustZone protections



As shown above, when TrustZone is enabled (TZEN = 0xB4), the application code can use the HDP area that is part of the flash memory watermark-based secure area. Indeed, when the application sets the HDPx_ACCDIS bit, data read, write, and instruction fetch on this HDP area is denied until the next system reset.

For example, the software code in the secure flash memory HDP area can be executed only once, with any further access to this area denied until the next system reset. Additionally, any flash memory page belonging to an active HDP area cannot be erased anymore.

When the TrustZone is disabled (TZEN = 0xC3), the volatile/non-volatile secure area features are deactivated and all secure registers are RAZ/WI.

See *Section 7: Embedded flash memory (FLASH)* for more details.

On-the-fly encryption/decryption (OTFDEC)

When the TrustZone security is enabled (TZEN = 0xB4), the OTFDEC can be initialized only by secure applications. Each of the four encrypted regions, once the configuration is confirmed, can be write-locked until the next power-on-reset.

Note: Any application (secure or nonsecure) can verify the initialization context of each OTFDEC region (including CRC of the keys), by reading the peripheral registers.

Key registers in each OTFDEC are write-only.

See *Section 3.10.3* for more details on this cryptographic engine.

Direct memory access controllers (GPDMAx)

When a DMA channel x is defined as secure (SECx = 1 in GPDMA_SECCFGR), the source and destination transfers can be independently set as secure or nonsecure by a secure application using SSEC and DSEC bits in GPDMA_CxTR1. *Table 11* summarizes these security options available in each DMA channel.

Table 11. DMA channel use (security)(1)

Destination type	Secure DMA channel x (SECx = 1)		Nonsecure DMA channel y (SECy = 0)	
	Secure source	Nonsecure source	Secure source	Nonsecure source
Secure destination	OK	OK(2)	Transfer blocked	
Nonsecure destination	OK(3)	OK(4)	Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return 0s. An illegal access event to TZIC is automatically triggered by the memory/peripheral used as source or destination.
2. If the source is a memory, the transfer is possible only if SSEC = 0, otherwise the transfer is blocked.
3. If the destination is a memory, the transfer is possible only if DSEC = 0, otherwise the transfer is blocked.
4. If the transfer is memory-to-memory, it is possible only if SSEC = 0 and DSEC = 0, otherwise it is blocked.

When a channel is configured as secure:

- Registers allocated to this channel (excluding GPDMA_SECCFGR, GPDMA_PRIVCFGR and GPDMA_RCFGLOCKR) are read as 0. Writes are ignored for nonsecure accesses. A secure illegal access event may also be triggered towards the TZIC peripheral.
  - Writes to GPDMA_SECCFGR and GPDMA_RCFGLOCKR must be secure. For each bit in GPDMA_PRIVCFGR, write must be secure if the corresponding bit in GPDMA_SECCFGR is set.
- In linked-list mode, the loading of the next linked-list data structure from memory is performed with secure transfers.
- When switching to a nonsecure state, the secure application must abort the channel or wait until the secure channel is completed before doing the switch.

Note: DMA secure channels are not available when TrustZone is deactivated. When a channel is configured as nonsecure, in linked-list mode, the loading of the next linked-list data structure from memory is performed with nonsecure transfers.

See *Section 16: General purpose direct memory access controller (GPDMA)* for more details.

Power control (PWR)

When the TrustZone security is enabled (TZEN = 0xB4), the selected PWR registers can be secured through PWR_SECCFGR, protecting the following PWR features:

- low power mode setup
- wake-up (WKUP) pins definition
- voltage detection and monitoring
- backup domain control

Other PWR configuration bits become secure:

- when the system clock selection is secure in the RCC: the voltage scaling (VOS) becomes secure.
- when a GPIO is configured as secure: its corresponding bit for pull-up/pull-down configuration in Standby mode becomes secure.
- when the USB Type-C/USB power delivery interface (UCPD) is configured as secure in TZSC: PWR_UCPDR register becomes secure.

See *Section 10: Power control (PWR)* for details.

Secure clock and reset (RCC)

When the TrustZone security is enabled (TZEN = 0xB4) and security is enabled in the RCC, the bits controlling the peripheral clocks and resets become TrustZone-aware:

- If the peripheral is securable and programmed as secure in the TZSC, the peripheral clock and reset bits become secure.
- If the peripheral is TrustZone-aware, the peripheral clock and reset bits become secure when at least one function is configured as secure inside the peripheral.

Note: Refer to Section 3.6.4 for the list of securable and TrustZone-aware peripherals.

Additionally, the following configurations can be made secure-only using RCC_SECCFGR:

- external clock (such as HSE or LSE), internal oscillator (such as HSI, CSI, or LSI)
- main PLL and AHB prescaler
- system clock source selection
- MCO clock output
- reset flag
- automatic internal oscillator waking up configuration

See *Section 11: Reset and clock control (RCC)* for details.

Real-time clock (RTC)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured RTC register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the RTC illegal access event is enabled in the TZIC.

After a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes. The secure boot code can then change the security setup, making registers Alarm A, Alarm B, wakeup timer, and timestamp secure or not, using RTC_SECCFGR.

When the SEC bit is set in secure-only RTC_SECCFGR:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER, and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When SEC is cleared in secure-only RTC_SECCFGR, it is still possible to restrict access in secure mode to some RTC registers by setting dedicated control bits: INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC, and ALRBSEC.

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

See *Section 46: Real-time clock (RTC)* for more details.

Tamper and backup registers (TAMP)

Like all TrustZone-aware peripherals, a nonsecure read/write access to a secured TAMP register is RAZ/WI. It also generates an illegal access event that triggers a secure illegal access interrupt if the TAMP illegal access event is enabled in the TZIC.

After a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes. The secure boot code can change this security setup, making some registers secure or not as needed, using TAMP_SECCFGR register.

When TAMPSEC is set in TAMP_SECCFGR:

- Writing the TAMP registers is possible only in secure mode. Backup registers have their own write protection (see below).
- Reading the TAMP registers (exception TAMP_SECCFGR, TAMP_PRIVCFGR, and TAMP_MISR) returns 0 if the access is nonsecure. Backup registers have their own read protection (see below).

The application can also:

- make TAMP_COUNTR register read and write secure-only by setting the CNT1SEC bit in TAMP_SECCFGR secure register
- increase security in backup registers for two of the three protection zones configured using BKPRWSEC[7:0] and BKPWSEC[7:0] bitfields in TAMP_SECCFGR:
  - protection zone 1 is read nonsecure, write nonsecure
  - protection zone 2 is read nonsecure, write secure
  - protection zone 3 is read secure, write secure

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

See *Section 47: Tamper and backup registers (TAMP)* for more details.

General-purpose I/Os (GPIO)

When the TrustZone security is enabled (TZEN = 0xB4), each I/O pin of the GPIO ports can be individually configured as secure through the GPIOx_SECCFGR registers. Only a secure application can write to GPIOx_SECCFGR registers. After boot, each I/O pin is set as secure.

When an I/O pin is configured as secure, its corresponding configuration bits for alternate function (AF), mode selection (MODE), and I/O data are RAZ/WI in case of nonsecure access.

When a digital alternate function is used (input/output mode) to protect data transiting from/to the I/O managed by a secure peripheral, the devices add a secure alternate function gate on the path between the peripheral and its allocated I/Os:

- if the peripheral is secure, the I/O pin must also be secure to allow input/output of data
- if the peripheral is not secure, the connection is allowed regardless of the I/O pin state.

The TrustZone-aware logic around GPIO ports, used as alternate functions, is summarized in *Table 12*.

Table 12. Secure alternate function between peripherals and allocated I/Os

Security configuration		Alternate function logic		Comment
Peripheral	**Allocated I/O pin**	**Input**	**Output**	
Secure	Secure	I/O data	Peripheral data	-
Nonsecure				Out of reset configuration
Secure	Nonsecure	0	0	-
Nonsecure		I/O data	Peripheral data	

When an analog function with an analog switch is used, the connection to the peripherals listed in *Table 13* is blocked by hardware when the peripheral is nonsecure and the I/O is secure.

Table 13. Nonsecure peripheral functions that cannot be connected to secure I/Os

Peripheral	Analog function(1)	Input	Output
ADC12	ADC12_INy (y = 1 to 17)	X	-

1. Used to find the I/O corresponding to the signal/function on the package (refer to the product datasheet).

Finally, regarding GPIO and security, *Table 14* summarizes the list of I/Os without any hardware protection linked to TrustZone. The listed signals (input and/or outputs) are not blocked when the I/O is set as secure, and the associated peripheral is non secure.

For example, when a secure application sets PA4 as secure to be used as LPTIM2_CH1, if the DAC is nonsecure, it can be programmed to output data to PA4, potentially causing malfunction to the secure application.

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

It is important that, for each case described in *Table 14*, the secure application decides if a potential effect on data integrity or confidentiality is critical or not. For example, if the USART situation described above is not acceptable (data transiting on secure USART is confidential), then the secure application must configure the TAMP as secure even if not used by the secure application.

Table 14. Nonsecure peripheral functions that can be connected to secure I/Os

Peripheral	Signal(1)	Input	Output	How to set the peripheral or function as secure
DAC	DAC1_OUTx (x = 1, 2)	-	X	Set DAC1SEC in GTZC1_TZSC_SECCFGR1.
UCPD	UCPD1_CCx (x = 1, 2)	X	X	Set UCPD1SEC in GTZC1_TZSC_ SECCFGR1.
	UCPD1_DBx (x = 1, 2)	X	-	
TAMP	TAMP_INx (x = 1 to 8)	X	-	Set TAMPSEC in TAMP_SECCFGR
	TAMP_OUTx (x = 1 to 8)	-	X	
RTC	RTC_OUTx (x = 1, 2)	-	X	Set SEC in RTC_SECCFGR.
	RTC_TS	X	-	Set TSSEC in RTC_SECCFGR.
PWR	WKUPx (x = 1 to 8)	X	-	Set WUPxSEC in PWR_SECCFGR.
RCC	LSCO	-	X	Set LSESEC in RCC_SECCFGR.
EXTI	EXTIx (x= 0 to 58)	X	-	Set SECx bit in EXTI_SECCFGR.

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

Refer to *Section 13: General-purpose I/Os (GPIO)* for more details.

Extended interrupts and event controller (EXTI)

When the TrustZone security is enabled, the EXTI is able to protect event register bits from being modified by nonsecure accesses. The protection can individually be activated per

input event via the register bits in EXTI_SECCFGRx. When an input event is configured as secure, only a secure application can change the configuration (including security), change the masking or clear the status of this input event.

The security configuration in EXTI_SECCFGR1 and EXTI_SECCFGR2 can be globally locked after reset in EXTI_LOCKR.

See *Section 18: Extended interrupts and event controller (EXTI)* for more details.

System configuration, boot, and security (SBS)

Like all TrustZone-aware peripherals, when the TrustZone security is enabled, a nonsecure read/write access to a secured SBS register is RAZ/WI. Such access also generates an illegal access event that triggers a secure illegal access interrupt if the SBS illegal access event is not masked in the TZIC.

See *Section 14: System configuration, boot, and security (SBS)* for more details.

Microcontroller debug unit (DBGMCU)

The MCU debug component (DBGMCU) helps the debugger, providing support for:

- low-power modes behavior during debug
- peripheral freeze during debug, applicable to I2Cs, IWDG, WWDG, timers, low-power timers, and GPDMA channels

The DBGCMU is a TrustZone-aware peripheral, managing accesses to its control registers as described in *Table 15*.

Table 15. TrustZone-aware DBGMCU nonsecure accesses management

Debug profile	Peripheral status(1)	DBG_xx_STOP control bits	
		Write access	Read access
Nonsecure invasive (SPIDEN = 0)	NS	Yes (S(2) or NS)	Yes (S or NS)
	S	None (S or NS)	
Secure invasive (SPIDEN =1)	NS	Yes (S or NS)	
	S	Yes (S only)	

1. As reported by the GTZC, the TrustZone-aware peripheral or the DMA channel.
2. Secure access from debugger is converted to nonsecure access in the device.

Refer to *Section 59.12: Microcontroller debug unit (DBGMCU)* for more details.

3.6.6 Activating TrustZone security

The TrustZone is deactivated by default in all devices. It can be activated by setting the TZEN option bit in FLASH_OPTSR2_PRG when PRODUCT_STATE is Open. Once TZEN has changed from disabled to enabled (0xC3 to 0xB4), the default security state, after reset, is always the following:

- CPU subsystem
  - Cortex-M33 exits reset in secure state, hence the boot address must point toward a secure memory area.
  - All interrupt sources are secure (in NVIC).
  - The memory mapped viewed by the CPU through IDAU/SAU is fully secure.

- Embedded flash memory
  - Flash memory nonvolatile secure areas (with their HDP zone), are defined with nonvolatile registers FLASH_SECWMxR (x = 1, 2). Default secure option bytes setup is all user flash secure, without HDP area defined.
  - Volatile block-based security attributions of the flash memory are nonsecure.
- Embedded SRAMs
  - All SRAMs are secure, as defined in GTZC/MPCBB (see *Section 3.6.4*). The secure boot code can change this security setup, making blocks secure or not.
- External memories
  - All memory devices connected to the FMC and OCTOSPI are secure, as defined in GTZC/MPCWM (see *Section 3.6.4*). The secure-boot code can change this security setup, making components secure or not.
- All GPIOs are secure.
- All GPDMA channels are nonsecure.
- Backup registers are nonsecure.
- Peripherals and GTZC:
  - Securable peripherals are nonsecure and unprivileged.
  - TrustZone-aware peripherals are nonsecure, with their secure configuration registers being secure.
  - All illegal access interrupts in GTZC/TZIC are disabled.

Note: Refer to Section 3.6.4 for the list of securable and TrustZone-aware peripherals.

3.6.7 Deactivating TrustZone security

Once TrustZone is activated, it can only be deactivated during a PRODUCT_STATE regression to Open.

Note: Such PRODUCT_STATE regression triggers the erasing of embedded memories (SRAM2, flash), and the reset of all peripherals, including the OTFDEC and all crypto engines.

After the TrustZone deactivation, most of the features mentioned in *Section 3.6* are no longer available:

- The nonvolatile secure area of the embedded flash memory is deactivated, including the HDP area.
- Only the NVIC manages nonsecure interrupts.
- All secure registers in TrustZone-aware peripherals are RAZ/WI.

Note: When the TrustZone is deactivated, the resource isolation using privilege remains available (see Section 3.7.2 for details).

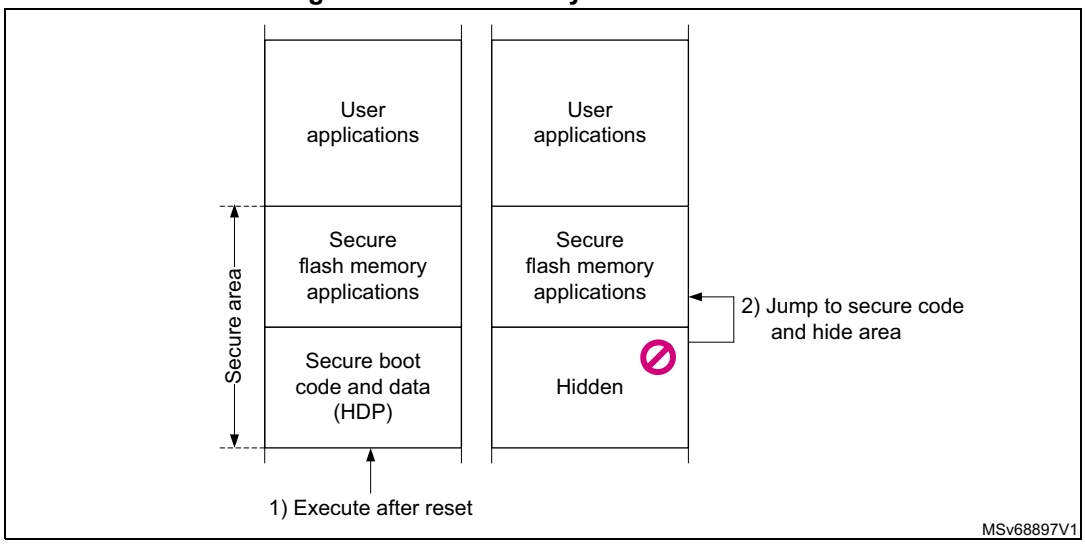
3.7 Other resources isolation

These are hardware mechanisms offering an additional level of isolation on top of the TrustZone technology.

3.7.1 Temporal isolation using secure hide protection (HDP)

The embedded flash memory allows to define an HDP area per watermarked-secure area of each bank (8-Kbyte page granularity). The code executed in this HDP area, with its related data and keys, can be hidden after boot, until the next system reset. The hide protection principle is shown in *Figure 9*.

Figure 9. Flash memory secure HDP area



Activation of HDP area in user flash memory is related to HDPL1: as soon as HDPL ≥ 2, data read, write, and instruction fetch (on the area defined by HDPx_PSTRT and HDPx_PEND in FLASH_HDPxR_PRG option bytes), are denied until the next device reset.

The END of the HDPx areas can be extended (dynamically by the application) thanks to FLASH_HDPEXTR flash memory register.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area).

3.7.2 Resource isolation using Cortex privileged mode

In parallel to the TrustZone isolation described in *Section 3.6*, the hardware and software resources can be partitioned so that they are restricted to software running in Cortex privileged mode.

Thanks to this hardware isolation technology, available even if TrustZone is disabled, critical code or data can be protected against intentional or unintentional tampering from the more exposed unprivileged code.

Memory and peripheral privileged allocation using MPU

The Cortex-M33 MPU divides the unified memory into eight regions in nonsecure and twelve in TrustZone, each aligned to a multiple of 32 bytes. Each memory region can be programmed to generate faults when accessed inappropriately by unprivileged software.

Memory and peripheral privileged allocation using GTZC

For the Cortex-M33 master, to complement the coarse isolation provided by the MPU, the GTZC reinforces, in a flexible way, the isolation between privileged and unprivileged tasks, for peripherals and selected memories.

For masters other than the Cortex-M33, the GTZC can assign them as unprivileged initiators, automatically protecting resources defined as privileged against this master.

- Securing peripherals with TZSC (privileged-only)

  In the devices, a peripheral is either securable privileged-only through GTZC, or is natively privileged-aware:

  – A securable privileged-only peripheral or memory is protected by an AHB/APB firewall gate controlled by the TZSC.

  – A privileged-aware peripheral or memory is connected directly to AHB or APB interconnect, implementing a specific behavior (for example, a subset of registers or a memory area is privilege-only).

  When such a peripheral is made privileged-only with GTZC, if it is master on the interconnect (SDMMC), it automatically issues privileged transactions. Privilege-aware masters like GPDMA1 and GPDMA2, drive privileged signal in the AHB interconnect according to their internal privileged mode, independently to the GTZC.

  The list of securable peripherals can be found in *Section 5: Global TrustZone® controller (GTZC)*.

- Securing memories with TZSC and MPCBB (privileged-only)

  The TZSC logic in GTZC provides the capability to manage the privilege level for all securable external memories, programming the MPCWM resources defined in *Section 3.6.4*.

  Similarly, the TZSC logic in GTZC provides the capability to configure the privilege level of embedded SRAM blocks, programming the MPCBB resources defined in *Section 3.6.4*.

- Error management (privileged-only)

  – Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal unprivileged read and write accesses.

  – The addressed resource follows a silent-fail behavior, returning all-0 data for read and ignoring any write.

  – When an illegal unprivileged access occurs, no bus error is generated, except when this is an instruction fetch, accessing a privileged memory or a peripheral register.

Managing security in privileged-aware peripherals

TrustZone-aware peripherals also implement privileged-only access mode. The privileged protection is valid even if TrustZone is disabled (TZEN = 0xC3):

- *Embedded flash memory*

  By default, all registers can be read or programmed in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in FLASH_PRIVCFGR, reading and writing the secure registers is possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0xC3).

  When nonsecure privileged bit NSPRIV is set in FLASH_PRIVCFGR, reading and writing the flash memory nonsecure registers is possible only in privileged mode.

  Regarding privileged protection, the devices offer the following features:

  – The system flash memory can be accessed in both privileged and unprivileged modes.
  – Each watermark-based secure area, including its secure HDP area, is accessible in secure-privileged and secure-unprivileged mode, if applicable.
  – Each 8-Kbyte page of the embedded flash memory can be programmed on-the-fly as privileged only, using the block-based privileged configuration registers FLASH_PRIV1BBRx and FLASH_PRIV2BBRx. An unprivileged page is accessible by privileged or unprivileged access.

Note: Switching a page from privileged to unprivileged does not erase its content.

When applicable, an erase or program operation is always available to privileged code, and is available to unprivileged code only for unprivileged pages or unprivileged memory.

- *On-the-fly encryption/decryption (OTFDEC)*

  When privileged bit PRIV is set in OTFDEC_PRIVCFGR, only a privileged application can initialize the OTFDEC.

Note: OTFDEC_PRIVCFGR can be read by both privileged and unprivileged code.

- *Direct memory access controllers (GPDMAx)*

  When a DMA channel x is defined as privileged (PRIVx = 1 in GPDMA_PRIVCFGR), special rules (see *Table 16*) apply when accessing privileged/unprivileged source or destination.

Table 16. DMA channel use (privilege)

Destination	Privileged DMA channel x (PRIVx = 1)		Unprivileged DMA channel y (PRIVy = 0)	
	Privileged source	Unprivileged source	Privileged source	Unprivileged source
Privileged	OK		Transfer blocked(1)	
Unprivileged			Transfer blocked	OK

1. When a transfer is blocked, the transfer completes but the corresponding writes are ignored, and reads return 0s.

- *Power control (PWR)*

  By default, after a power-on or a system reset, all PWR registers except PWR_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in PWR_PRIVCFGR, reading and writing the PWR securable registers are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0xC3).

  When nonsecure privileged bit NSPRIV is set in PWR_PRIVCFGR, reading and writing the PWR nonsecure registers are possible only in privileged mode.

  See *Section 10: Power control (PWR)* for details.

- *Secure clock and reset (RCC)*

  By default, after a power-on or a system reset, all RCC registers except RCC_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When secure privileged bit SPRIV is set in RCC_PRIVCFGR, reading and writing the RCC securable bits are possible only in privileged mode. Write access to this bit is ignored if TrustZone is disabled (TZEN = 0xC3).

  When nonsecure privileged bit NSPRIV is set in RCC_PRIVCFGR, reading and writing the RCC nonsecure bits are possible only in privileged mode.

  See *Section 11: Reset and clock control (RCC)* for details.

- *Real time clock (RTC)*

  By default after any backup domain reset, all RTC registers except RTC_PRIVCFGR, can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only RTC_PRIVCFGR:

  – Writing the RTC registers is possible only in privileged mode.

  – Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privileged and unprivileged modes.

  All the other RTC registers can be read only in privileged mode.

  When PRIV bit is cleared in privileged-only RTC_PRIVCFGR register, it is still possible to restrict access to privileged mode to some RTC registers by setting dedicated control bits: INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV, or ALRBPRIV.

  See *Section 46: Real-time clock (RTC)* for details.

- *Tamper and backup registers (TAMP)*

  By default after any backup domain reset, all TAMP registers except TAMP_PRIVCFGR can be read or written in both privileged and unprivileged modes.

  When PRIV bit is set in privileged-only TAMP_PRIVCFGR:

  – Writing the TAMP registers is possible only in privileged mode, except for the backup registers and the monotonic counters that have their own protection setting.

  – Reading the TAMP_SECCFGR or TAMP_PRIVCFGR is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privilege mode, except for the backup registers and the monotonic counters that have their own protection setting.

The application can also:

– make TAMP_COUNT1R register read and write privileged-only by setting the CNTPRIV bit in TAMP_PRIVCFGR

– increase security for two of the three protection zones in backup registers, using BKPRWPRIV and BKPWPRIV bits in TAMP_PRIVCFGR:

– Make protection zone 1 read privileged, write privileged.

– Make protection zone 2 read privileged or unprivileged, write privileged.

– Protection zone 3 is always read and write privileged or unprivileged.

- *General-purpose I/Os (GPIO)*

  All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or nonsecure).

- *Extended interrupts and event controller (EXTI)*

  The EXTI peripheral is able to protect event register bits from being modified by unprivileged accesses. The protection is individually activated per input event via the register bits in the privileged-only EXTI_PRIVCFGRx registers. When an input event is configured as privileged, only a privileged application can change the configuration (including security if applicable), change the masking or clear the status of this input event.

  The security configuration in EXTI_PRIVCFGR1 and EXTI_PRIVCFGR2 can be globally locked after reset in EXTI_LOCKR.

  See *Section 18: Extended interrupts and event controller (EXTI)* for more details.

- *System configuration boot and security (SBS)*

  All SBS registers can be read and written in both privileged and unprivileged modes, except:

  – FPUSEC bit in SBS_SECCFGR registers (privileged only)

  – SBS registers for CPU configuration: SBS _CSLOCKR, SBS _FPUIMR, and SBS _CNSLCKR

  See *Section 14: System configuration, boot, and security (SBS)* for more details.

3.8 Secure execution

Through a mix of special software and hardware features, the devices ensure the correct operation of their functions against abnormal situations caused by programmer errors, software attacks through network access or local attempt for tampering code execution.

This section describes the hardware features specifically designed for secure execution.

3.8.1 Memory protection unit (MPU)

The Cortex-M33 includes a memory protection unit (MPU) that can restrict the read and write accesses to Each memory region (including those mapped to peripherals), based on one or more of the following parameters

- Cortex-M33 operating mode (privileged, unprivileged)
- data/instruction fetch

The memory map and the programming of the nonsecure and secure MPUs split memory into regions (up to eight for the nonsecure, and up to twelve for the TrustZone). Secure MPU is only available when TrustZone is activated.

3.8.2 Embedded flash memory write protection

The embedded flash memory write protection (WRP) prevents illegal or unwanted write/erase to special sections of the embedded flash memory user area (system area is permanently write protected).

Write protected area is defined through the option bytes, writing the start and end addresses: two write-protected areas can be defined in each bank, with the granularity of a 32-Kbyte page.

WRP areas can be modified through option byte changes unless the corresponding FLASH_WRPSGNxR has its UNLOCK option bit cleared (meaning ROM emulation). UNLOCK can be set only when regressing to PRODUCT_STATE = Open.

Note: Bank erase aborts when it contains a write-protected area (WRP or HDP area).

3.8.3 Tamper detection and response

Principle

The devices include active protection of critical security assets against temperature, voltage and frequency attacks, with the following features:

- erasure of device secrets upon tamper detection
- improved guarantee of safe execution for the CPU and its associated security peripherals, including:
  - out-of-range voltage (example: V_{BAT}, V_{DDA}), temperature and clocking (LSE) detection
  - security watchdog IWDG clocked by the internal oscillator LSI
  - possible selection of internal oscillator HSI as system clock
- power supply protection
  - RTC/TAMP domain powered automatically with V_{DD} or V_{BAT}

See *Section 47: Tamper and backup registers (TAMP)* for more details.

Tamper detection sources

The devices support eight active input/output pins, allowing four independent active-tamper meshes, or up to seven meshes if the same output pin is shared by several input pins (for a total of eight active-tamper I/Os). The active-tamper balls are mapped in the center of packages that can be used in the POS market (such as WLCSP80).

The active pins are clocked by the LSE, and are functional in different system operating modes (Run, Sleep, Stop, or Standby), and in VBAT mode. Refer to TAMP pins functionality over modes for list of tamper pins and their availability across power modes

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

Note: Timestamps are automatically generated when a tamper event occurs.

The internal tamper sources are listed in *Table 17*.

Table 17. Internal tampers in TAMP

Tamper input	NOER bit number in TAMP_CR3	Tamper source
itamp1	0	Backup domain voltage continuous monitoring, functional in VBAT mode
itamp2	1	Temperature monitoring, functional in VBAT mode
itamp3	2	LSE monitoring(1), functional in VBAT mode
itamp4	3	HSE monitoring
itamp5	4	RTC calendar overflow (rtc_calovf)
itamp6	5	JTAG/SWD access
itamp7, 12, 13	6, 11, 12	Voltage monitoring (V_{CORE}, V_{REF+}), through ADC analog watchdog
itamp8	7	Monotonic counter overflow (generated internally)
itamp9	8	Fault generation for cryptographic peripherals (SAES, PKA, AES, RNG)
itamp11	10	IWDG timeout and potential tamper (IWDG reset when at least one enabled tamper flag is set)
itamp15	14	System fault detection

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

Response to tampers

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

- Generate an interrupt, capable of waking up the device from Stop and Standby modes (see TAMPxMSK bits in TAMP_CR2 register).
- Generate a hardware trigger for the low-power timers.
- Erase device secrets if the corresponding TAMPxNOER bit is cleared in TAMP_CR2 (for tamper pins) or TAMP_CR3 (for internal tamper). These erasable secrets are:
  - symmetric keys stored in backup registers (x32), in AES, HASH, and OTFDEC (encrypted flash memory regions are read as 0)
  - asymmetric keys stored in PKA SRAM, erased when V_{DD} is present
  - other secrets stored in SRAM2 and CPU instruction cache memory (SRAM2 erased when V_{DD} is present)
  - nonvolatile information used to derive the DHUK in SAES is 0-ed until complete SRAM2 erase
  - Backup SRAM (depending on configuration bit), erased when V_{DD} is present

Read/write accesses by software to all these secrets can be blocked, by setting the BKBLOCK bit in TAMP_CR2. The device secrets access is possible only when BKBLOCK is cleared, and no tamper flag is set for any enabled tamper source.

If V_{DD} is not present, the secrets that are erased when V_{DD} is present are erased only at the next V_{DD} power on.

Note: Device secret erase is also triggered by setting the BKERASE bit in TAMP_CR2, or by performing a PRODUCT_STATE regression as defined in Section 3.11.2.

Device secrets are not reset by a system reset or when the device wakes up from Standby mode.

Software filtering mechanism

Each tamper source can be configured not to launch an immediate erase, by setting the corresponding TAMPxNOER bit in TAMP_CR2 (for external tamper pin) or TAMP_CR3 (for internal tamper).

In such situation, when the tamper flag is raised, access to below secrets is blocked until all tamper flags are cleared:

- DHUK in SAES: fixed to a dummy value
- Backup registers, backup SRAM, SRAM2: read as 0, write-ignored
- AES, SAES, and HASH peripherals: automatically reset by RCC
- PKA peripheral: reset, with memory use blocked (meaning PKA not usable)

Once the application, notified by the tamper event, analyzes the situation, there are two possible cases:

- The application launches secrets erase with a software command (confirmed tamper).
- The application clears the flags to release secrets blocking (false tamper).

Note: If the tamper software fails to react to such a tamper flag, an IWDG reset triggers an automatic erasing of secrets.

Tamper detection and low-power modes

The effect of low-power modes on a tamper detection are summarized in *Table 18*.

Table 18. Effect of low-power modes on TAMP

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

3.9 Secure storage

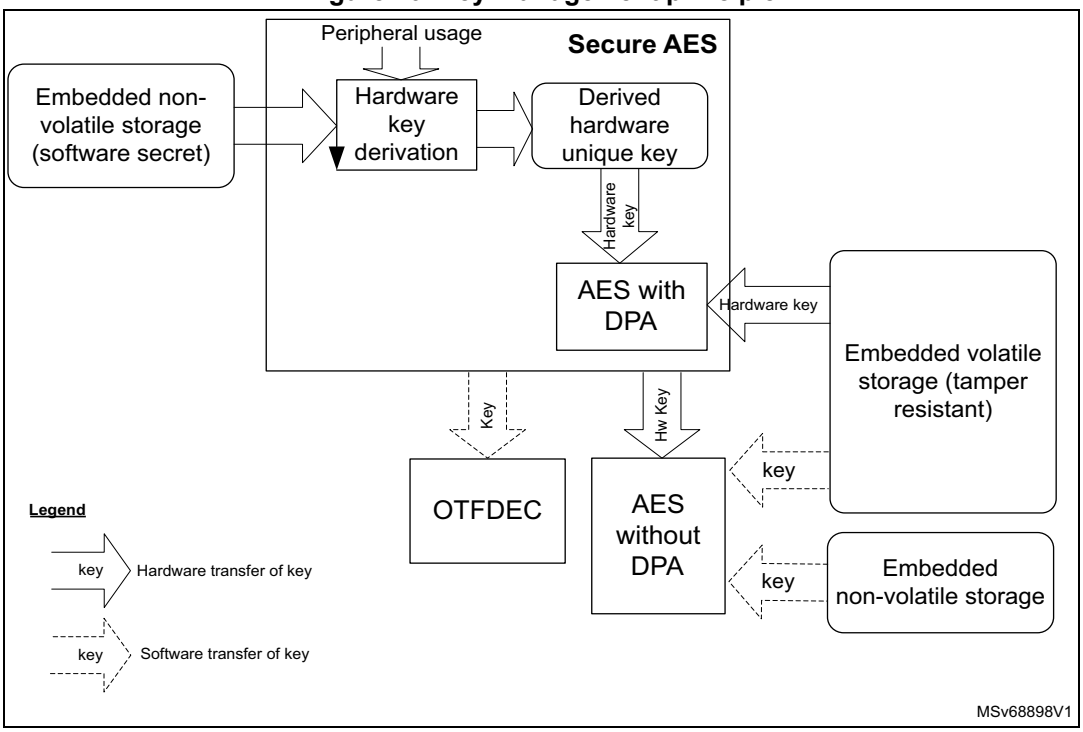
A critical feature of security systems is how long-term keys are stored, protected, and provisioned. Such keys are typically used for loading a boot image, or handling of critical user data.

Figure 10 shows how key management service application can use the AES engine, for example, to compute external image decryption keys. Non-volatile keys are stored in the flash memory in a dedicated area including access control (see *Section 3.7.1*), while volatile

key storage consists in the battery-powered, tamper-protected SRAM or registers in TrustZone-aware TAMP.

Figure 10 also shows keys that are manipulated by software (like OTFDEC keys), or keys that are managed only by hardware (like DHUK). More information on those hardware keys can be found in *Section 3.9.1*.

Figure 10. Key management principle



Details on tamper protection is found in *Section 3.8.3*, while TAMP TrustZone features are briefly described in *Section 3.6.5*.

3.9.1 Hardware secret key management

As shown in the previous figure, the devices propose a better protection for application keys, using hardware secret keys. These AES keys can be made usable to the application, without exposing them in clear-text (unencrypted). Such keys also become immediately unusable in case of tamper.

There are three different sources of hardware secret keys:

- DHUK: derived keys based on 256-bit nonvolatile device unique secret in flash memory. The flash memory provides value provisioned during product manufacturing (called RHUK). The generation of DHUK key takes into account the TrustZone state, the OBK-HDPLx (temporal isolation counter), the EPOCH (regression counter allowing antireplay protection) and key use state (KMOD).
- BHK: 256-bit application key stored in tamper-resistant volatile storage in TAMP

  This key is written at boot time, then read/write locked to the application until the next reset.
- XORK: result of an XOR of BHK and DHUK

Those keys can be used in the following modes:

- as a normal key, loading in write-only key registers (software key mode)
- as an encryption/decryption key for another key, to be used in the DPA-resistant SAES (wrapped key mode)
- as an encryption/decryption key for another key, to be used in a faster AES engine (shared key mode)

3.9.2 Unique ID

The devices store a 96-bit ID that is unique to each device (see *Section 60.1: Unique device ID register (96 bits)*).

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

Alternatively, the 256-bit device unique key (DHUK) can be used (see *Section 3.9.1*).

3.10 Crypto engines

The devices implement state-of-the-art cryptographic algorithms featuring key sizes and computing protection as recommended by national security agencies such as NIST for the U.S.A, BSI for Germany or ANSSI for France. Those algorithms are used to support privacy, authentication, integrity, entropy and identity attestation.

The embedded crypto engines reduce weaknesses on the implementation of critical cryptographic functions, preventing, for example, the use of weak cryptographic algorithms and key sizes. They also enable lower processing times and lower power consumption when performing cryptographic operations, offloading those computations from Cortex-M33. This is especially true for asymmetric cryptography.

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

For more information on crypto engine processing times, refer to their respective sections in the reference manual.

3.10.1 Crypto engines features

Table 19 lists the accelerated cryptographic operations available in the devices. Two AES accelerators are available (both can be reserved to secure application only).

Note: Additional operations can be added using firmware. The PKA can accelerate asymmetric crypto operations (like key pair generation, ECC scalar multiplication, point on curve check). See Section 36: Public key accelerator (PKA) for more details.

Table 19. Accelerated cryptographic operations

Operations	Algorithm	Specification	Length of keys (in bit)	Modes
Get entropy	RNG	NIST SP800-90B(1)	N/A	Software and hardware(2) modes running in parallel
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR(3)
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D	128, 256	GCM, CCM
Cipher-based message authentication code		NIST SP800-38D	128, 256	GMAC
Checksum	SHA-1	FIPS PUB 180-4	N/A	Digest 160-bit
Cryptographic hash	SHA-2		-	SHA-224, SHA-256 SHA2-384, SHA2-512
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	Short, long (>64 bytes)	-
Encryption/decryption key-pair generation(4)	RSA	IETF RFC 8017 NIST SP800-56B	Up to 4160	RSAES-OAEP
Signature(4) with hashing	RSA	IETF RFC 8017 FIPS PUB 186-4	Up to 4160	PKCS1-v1_5, PSS
Signature verification	ECDSA	ANSI X9.62 IETF RFC 7027 FIPS PUB 186-4	Up to 640	Refer to table 'Family of supported curves for ECC operations' in PKA section for details
Key agreement	ECDH	ANSI X9.42		

1. Certifiable using STMicroelectronics reviewed documents.
2. Random numbers distribution to SAES and PKA using a dedicated hardware bus.
3. ECB and CBC chaining modes protected against side-channel and timing attacks in SAES (see *Section 3.10.2*).
4. Private key cryptography protected against side-channel and timing attacks.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

3.10.2 Secure AES co-processor (SAES)

The devices provide an additional on-chip hardware AES encryption and decryption engine, implementing countermeasures and mitigations against power and electromagnetic side-channel attacks.

Clocked by the system clock the SAES results in very good performance for a dpa resistant hardware accelerator. The SAES engine supports 128-bit or 256-bit key in electronic code book (ECB), cipher block chaining (CBC), (CTR), (GCM), (CCM), (GMAC) modes.

As shown in *Section 3.9*, the SAES can be used for extra-secure on-chip storage for sensitive information. It can also be made secure-only.

For more information, refer to *Section 34: Secure AES coprocessor (SAES)*.

3.10.3 On-the-fly decryption engine (OTFDEC)

The OTFDEC TrustZone-aware peripheral proposes on-the-fly decryption of encrypted images stored on external flash memory, connected through the OCTOSPI. This decryption process introduces almost no additional cycle overhead when the standard NOR flash memory is used. The OTFDEC can also be used to encrypt flash memory images on the device (for example to encrypt with a device unique secret key).

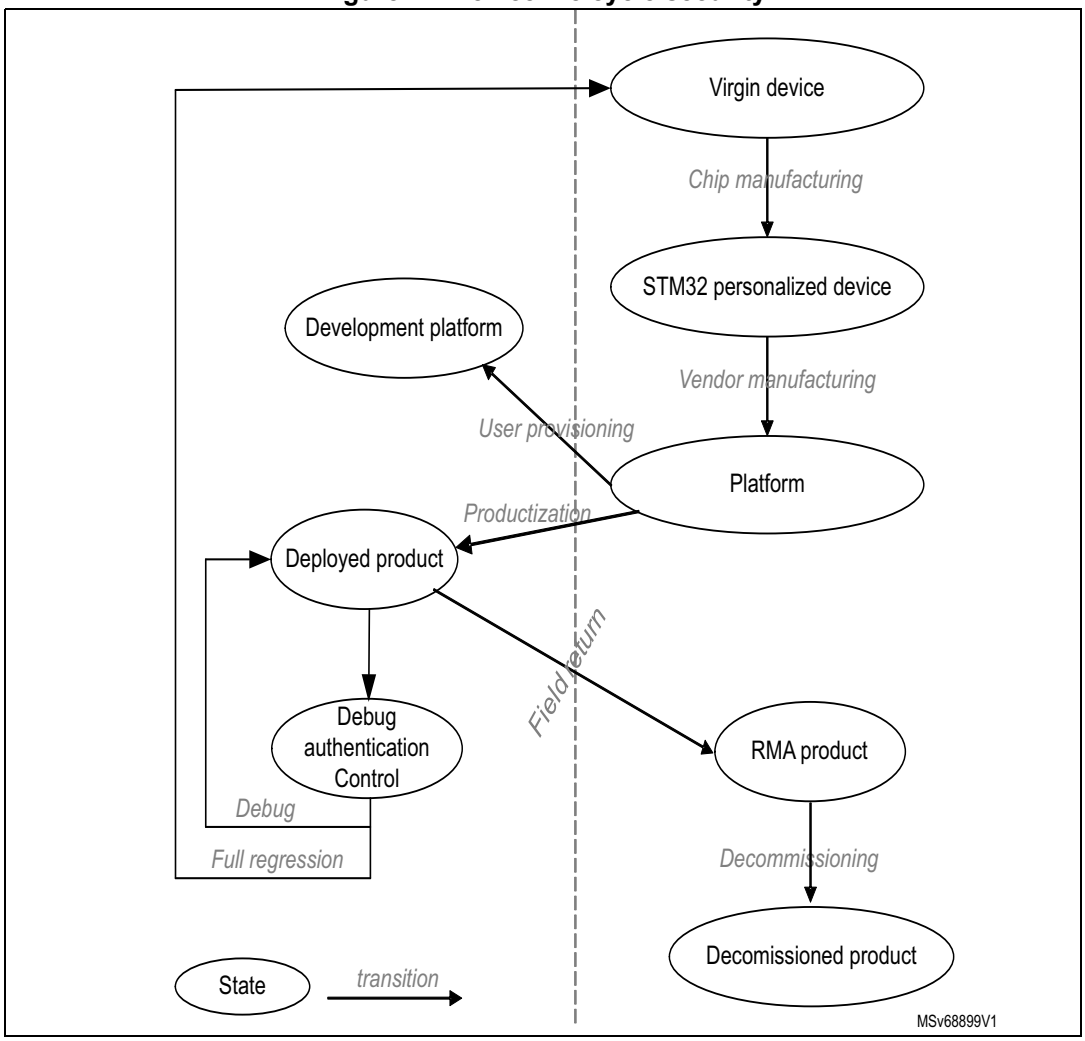
When a tamper event is confirmed in TAMP, all OTFDEC keys are erased and encrypted regions are read as 0 until the OTFDEC is properly initialized again.

An OTFDEC typical use is detailed in *Section 3.12.2*. For more details on the peripheral programming, refer to *Section 37: On-the-fly decryption engine (OTFDEC)*.

3.11 Product life cycle

A typical IoT device life cycle is summarized in *Figure 11*. For each step, the devices propose secure life cycle management mechanisms embedded in the hardware.

Figure 11. Device life cycle security



Additional details on the various phases and associated transitions, found either at the vendor or end-user premises, are summarized in *Table 20*.

Table 20. Main product life cycle transitions

Transitions	Description
Device manufacturing	ST creates new STM32 devices, always checking for manufacturing defects. During this process, the device is provisioned with ROM firmware, secure firmware install (SFI) unique key pair, and a public ID.
Vendor manufacturing	One (or more) vendor is responsible for the platform assembly, initialization, and provisioning before delivery to the end user. This end user can use the final product (“productization” transition) or he/she can use the platform for software development (“user provisioning” transition).
Productization	The end user gets a product ready for use. All security functions of the platform are enabled, the debugging/testing features are restricted/disabled, and unique boot entry to immutable code is enforced.
User provisioning	Platform vendor prepares an individual platform for development, not to be connected to a production cloud network.
Field return	The product is returned for analysis to field return centers. Analysis is possible by opening it partially, or by launching a partial or a full regression. Such accesses require provisioning, including certificates provided by the vendor.

The features described hereafter contribute to secure the device life cycle.

3.11.1 Product configurations and security services

The product is provisioned with ST security services in system flash, but can be configured to let the full control of the boot chain to OEMs. This is done thanks to the unique boot entry (FLASH_OPTSR: BOOT_UBE) option byte, allowing to select between the security services (ST-iROT: ST immutable root of trust), or OEM boot implementation (OEM-iROT: OEM immutable root of trust) to be installed in the user part of the flash memory.

Security services are provisioned by ST in system flash (immutable). They provide the root of trust of the platform managing the verification and the update of the first updatable code (uROT).

3.11.2 Life cycle management

The product life cycle allows to control access to different assets (code and data) of the product, including during development, manufacturing, and after sales. It allows to provision the product with different distribution models, taking care of the code and data provisioned.

Table 21. Typical product life cycle phases

PRODUCT_STATE + DebugState		Debug (default configuration)	Comments
Open	Device open	Secure(1) and nonsecure	Allows to develop the product, as it provides the debug of the code. Boot address must target a secure area when TrustZone is enabled. Using the boot pin allows to launch the bootloader. Using boot pin and BOOT_UBE allows to launch user flash code through ST-iROT.
Provisioning	Debug partially opened (only nonsecure)	HDPL3 + Nonsecure only	Allows to manage the provisioning of the product (partial or full). It allows to launch secure firmware Install, or bootloader to provision the product. Boot address must target a secure area when TrustZone is enabled. Boot on SRAM is not permitted.
iROT-Provisioned	Debug partially opened (only nonsecure)	HDPL3 + Nonsecure only	Assumes that immutable root of trust is installed, including its configuration (code, option bytes, secure storage). Boot address must target a secure area when TrustZone is enabled. Boot on SRAM is not permitted.
TZ-Closed	Debug partially opened (only nonsecure)	HDPL3 + Nonsecure only	Assumes that the Secure OS is installed in TrustZone (this state exist only if TZEN = 0xB4). Debug is opened for nonsecure applications
Closed	Debug closed	NoDebug + debug authentication control	Assumes the product configuration as finalized. It allows to support debug authentication for in-the-field repair (read the dedicated application note).
Locked	Debug locked	None	Assumes the product configuration as finalized. The debug authentication is not permitted. The product is definitively in this state.
NS-Regression	Debug closed	NoDebug	This is a temporal (but nonvolatile) state to manage the partial regression to the TZ-Closed, removing all nonsecure code and data including in secure storage.
Regression	Debug closed	NoDebug	This is a temporal state (but nonvolatile) to manage the full regression to Open state, removing all user flash code and data including in secure storage (HDPL1 to HDPL3).
Debug Constrained	Debug opened following debug authentication permissions	Depends upon permissions	This is a temporal state (until power-on-reset) to manage debug based on the permissions acquired by the debug authentication protocol.

1. Debug is not available when executing RSS code.

STMicroelectrincs does not perform analysis on defective parts in closed or locked states.

The supported transitions, summarized in *Figure 12*, can be requested (when available) through the debug interface or via the system bootloader.

Figure 12. PRODUCT_STATEs (simplified TrustZone activated view)

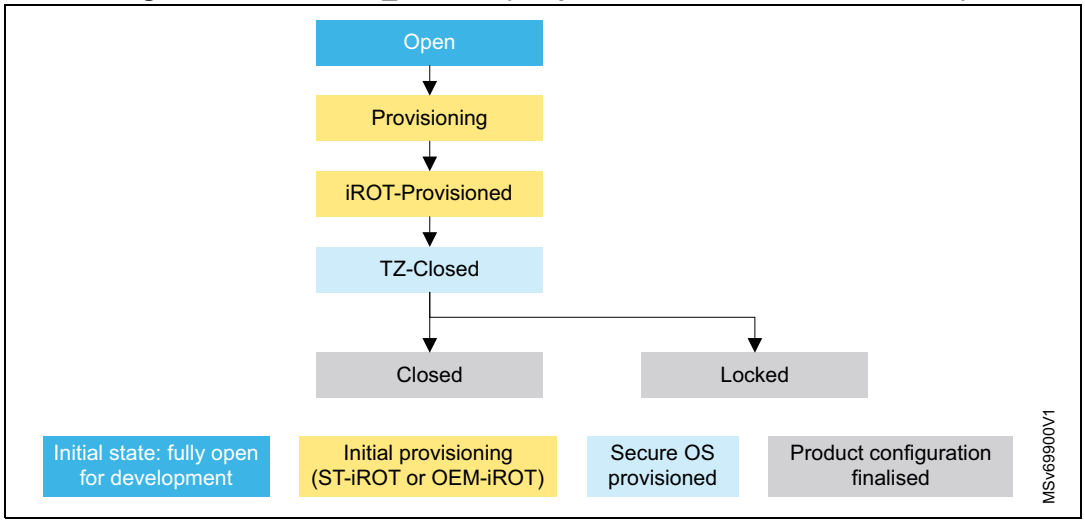
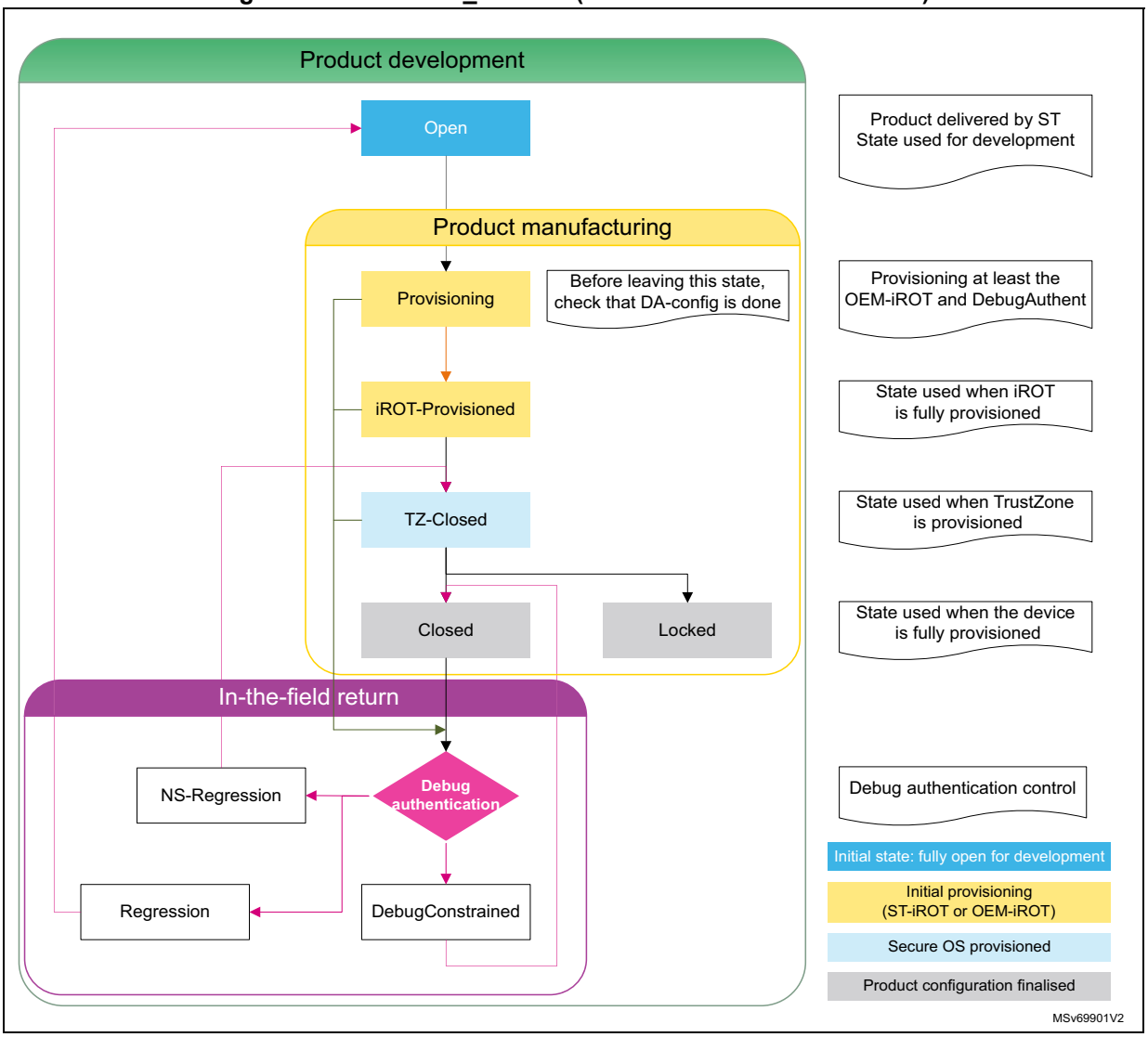


Figure 13. PRODUCT_STATEs (full TrustZone activated view)



The full view of the life cycle includes the debug authentication part, to help product maintenance to manage field returns (in-the-field centers). The debug authentication control is ensured thanks to a protocol based on Arm PSA ADAC specification. It allows controlling that the host has a trusted certificate with permissions. Permissions definition allows a lot of flexibility, in our default model we consider: Full or Partial regression, or to open the debug for nonsecure.

When TrustZone is disabled, the state TZ-Closed does not exist, and the debug authentication is based on a password authentication method. For details on the debug authentication feature, refer to the corresponding application note.

The procotol includes a “DISCOVER” command allowing to get basic informations of the target: manufacturer, product name, device identifier, protocol version supported and crypto scheme for authentication (this help selecting between certificates and password).

3.11.3 Recommended product settings

To ease the product maintenance (in-the-field), we recommend to take benefit of the feature called debug authentication control. This enables the maintenance of product activating the debug and makes possible to manage regressions while considering the security of the sensible information.

This implies the following actions:

- set the PRODUCT_STATE in Closed state when the product has been configured
- When debug authentication control is based on certificates (when TrustZone enabled), **provision the OEM-PublicKey and the SOC-Mask**. Setting the SOC-Mask to propose only the nonsecure-debug and partial and full regression guaranty to be compliant with the Arm PSA security model.
- When debug authentication control is based on password (when TrustZone is disabled), only the HASH of the password must be provisioned.

A dedicated application note is available to help configuring the platform, the host tools and to generate certificates.

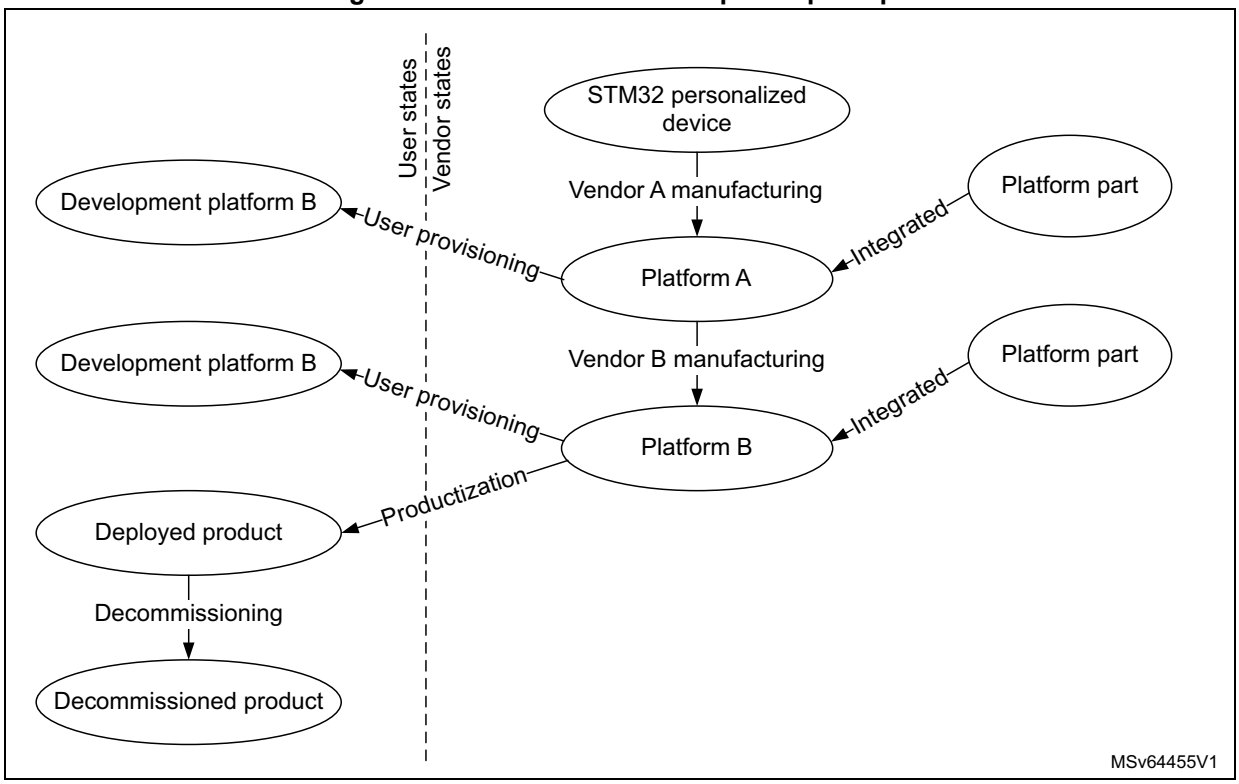
3.12 Software intellectual property protection and collaborative development

Thanks to software intellectual property protection and collaborative model, the devices allow the design of solutions integrating innovative third-party libraries.

Collaborative development is summarized in *Figure 14*. Starting from a personalized device sold by STMicroelectronics, a vendor A can integrate a portion of hardware and software on a platform A, which can then be used by a vendor B, who does the same before deploying a final product to the end users.

Note: Each platform vendor can provision individual platforms for development not to be connected to a production cloud network (“Development Platform X”).

Figure 14. Collaborative development principle



The features described hereafter contribute to securing the software intellectual property within such a collaborative development.

3.12.1 Software intellectual property protection

As described in *Section 3.11.2,* the hardware PRODUCT_STATE mechanism automatically controls the accesses to secrets provisioned in the device. The protection of these secrets are defined in *Table 22*.

Table 22. Software intellectual property protection with PRODUCT_STAT

PRODUCT_STATE protection level		Secrets protection
Open	Device open	No special protections.
Provisioning	Debug partially opened (only nonsecure)	All areas protected with HDPL1 cannot be dumped, debugged or traced. The iROT can setup a higher level of protection.
iROT-Provisioned	Debug partially opened (only nonsecure)	All areas protected with HDPL1 cannot be dumped, debugged or traced. The iROT can setup a higher level of protection.
TZ-Closed	Debug partially opened (only nonsecure)	All peripherals and memories mapped as secure during secure boot cannot be dumped, debugged or traced
Closed	No debug	No debug possible except through debug authentication.
Locked	No debug	All data and code stored in the device or encrypted in external flash memory cannot be dumped clear-text, debugged or traced.

3.12.2 Software intellectual property protection with OTFDEC

As described in *Section 3.10.3,* the OTFDEC associated with the OCTOSPI is able to decrypt on the fly, the encrypted images stored in external SPI flash memories.

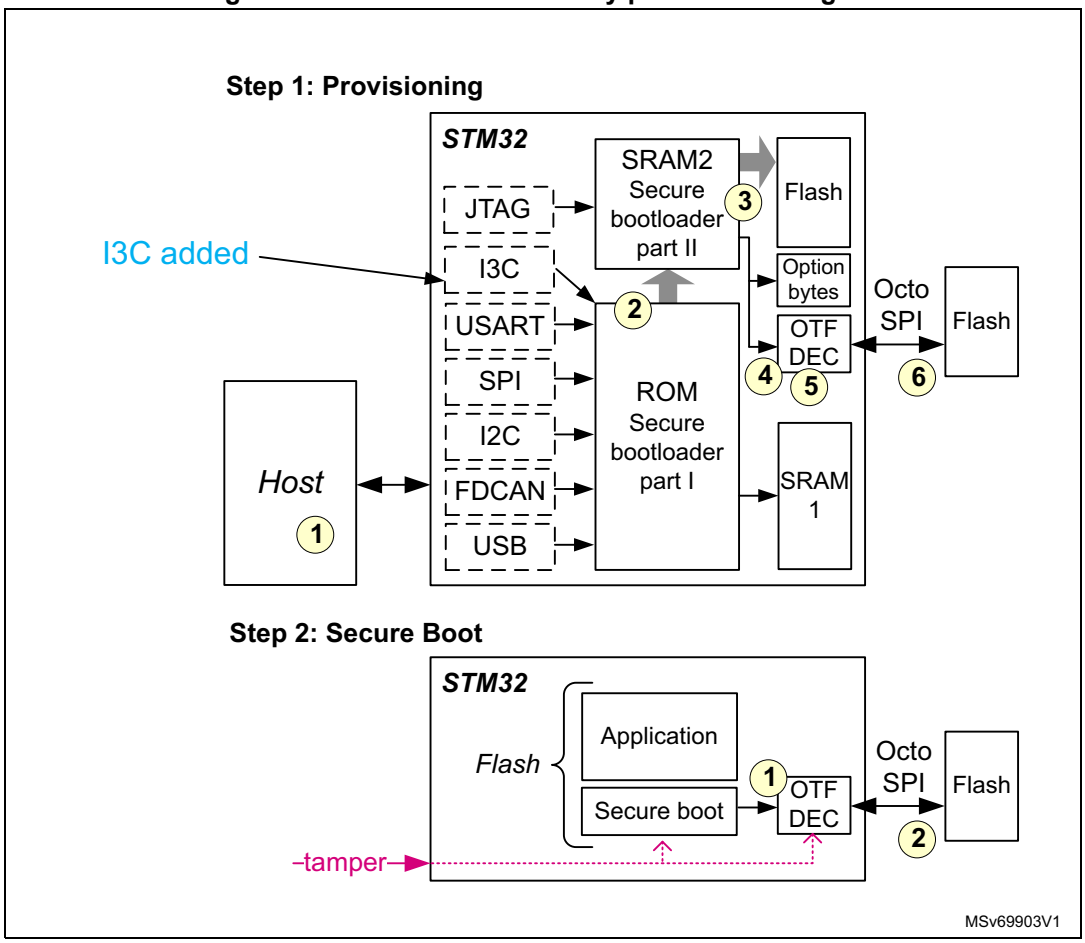
Thanks to this feature, the devices allow the installation of intellectual properties, in one of the following ways:

- over the air, with the image already encrypted with a key provisioned in the device
- through a provisioning host located in a trusted or a non-trusted environment/facility

Figure 15 illustrates this last case, with the provisioning, in a non-trusted environment, of software intellectual properties both in the embedded flash memory and in an external SPI flash memory (encrypted).

Note: Since the OTFDEC uses the AES in counter mode (CTR) to achieve the lowest possible latency, each time the content of one encrypted region is changed, the corresponding cryptographic context (key or initialization vector) must be changed. This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR flash memory.

Figure 15. External flash memory protection using SFI



Provisioning

Assuming the device is virgin, the first step is to provision both flash memories, as detailed below:

1. The user creates a SFI image, composed of:
   - encrypted internal firmware and data (including external flash memory drivers)
   - encrypted external firmware and data AES key (up to 4)
   - encrypted external firmware and data image
2. The secure bootloader stored in the system memory, loads the second part of the secure bootloader in SRAM2, through the supported communication ports (USART, SPI, I2C, FDCAN, USB, I3C, and JTAG). This second part runs in the secure SRAM2 and is responsible for executing the SFI process, applying the SFI protocol thanks to the commands received through the above mentioned supported communication ports.
3. The internal flash memory is programmed with decrypted option bytes, internal firmware and data, and external firmware and data AES keys. Alternatively, device unique external firmware AES keys can be used instead of such global keys.
4. The OTFDEC is properly initialized with encrypted region information, including the corresponding external firmware and data AES key.
5. Running the SFI process, chunks of encrypted external firmware and data image are decrypted in the device, then re-encrypted in the OTFDEC.

6. After a chunk OTFDEC re-encryption, the user external flash memory programmer is responsible for programming the last encrypted chunks to the external SPI flash memories through the OCTOSPI.

Secure boot

After provisioning, each time the device initializes on a trusted firmware, the following actions are required:

1. Secure-boot firmware executes, programming the external firmware and data AES keys to the OTFDEC write-only key registers, along with the other needed information.
2. The application reads or executes the encrypted external firmware and data through the OCTOSPI in memory mapped mode, unless a tamper event is detected. In this case, all OTFDEC keys are erased and encrypted regions are read as 0 until the OTFDEC is properly initialized.

For more information on SFI solutions for the devices, refer to the AN4992 "*STM32 MCUs secure firmware install (SFI) overview".*

3.12.3 Other software intellectual property protections

The device additional protections to software intellectual property are:

- Invasive attacks such as physical tampering or perturbation are countered by detection then decommissioning of the device before the detected attack succeeds.
- Noninvasive attacks, such as side channel attacks, are countered by not leaking secret information via side channels (such as timing, power, and EM emissions).

4 Boot modes

At startup, a BOOT0 pin and NSBOOTADD[31:8]/SECBOOTADD[31:8] option bytes are used to select the boot memory address that includes:

- Boot from any address in user flash memory
- Boot from system memory:
  - Bootloader
  - ST immutable root of trust (ST-iROT)
  - Root security service (RSS)
  - Debug authentication library (RSS-DA)

Embedded bootloader

The embedded bootloader is located in the system memory, programmed by ST during production. It is used to reprogram the flash memory by using USART, I2C, I3C, SPI, FDCAN, or USB_FS in device mode through the DFU (device firmware upgrade).

Refer to AN2606 "*STM32 microcontroller system memory boot mode*".

Embedded root security services (RSS)

The embedded RSS are located in the secure information block, programmed by ST during production.

Refer to AN4992 "*Overview secure firmware install (SFI)*".

Embedded immutable root of trust (ST-iROT)

The embedded ST-iROT in the system memory, is programmed by ST during production. ST-iROT is the immutable root of trust managing the secure boot and secure install of the first updatable level to execute in a boot sequence.

Refer to AN6007 "*Introduction to ST immutable Root of Trust (STiROT) for STM32H5 MCUs*".

Embedded debug authentication (ST-DA)

The embedded ST-DA in the system memory, is programmed by ST during production. ST-DA is the library that manages the debug authentication protocol by allowing to securely reopen the debug or to launch regressions on secured products in the field.

Refer to AN6008 "*Introduction to Debug Authentication (DA) for STM32 MCUs*".

4.1 STM32H523/62/63xx boot modes

Table 23 and *Table 24* provide the detail of the boot mode when the TrustZone is, respectively, disabled (TZEN = 0xC3) and enabled (TZEN = 0xB4).

Table 23. Boot mode when TrustZone is disabled (TZEN = 0xC3) - STM32H523/62/63xx devices

PRODUCT_STATE	BOOT0 pin	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	NSBOOTADD[31:8]	Boot address defined by user option byte NSBOOTADD[31:8]	Flash: 0x0800 0000
-	1	NA	Bootloader	Bootloader
Provisioning	x	NA	RSS	RSS
Provisioned, Closed, Locked	x	NSBOOTADD[31:8]	Boot address defined by user option byte NSBOOTADD[31:8]	Flash: 0x0800 0000

Table 24. Boot mode when TrustZone is enabled (TZEN = 0xB4) - STM32H523/62/63xx devices

PRODUCT_STATE	BOOT0 pin	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	SECBOOTADD[31:8]	Boot address defined by user option byte SECBOOTADD[31:8]	Flash: 0x0C00 0000
-	1	NA	Bootloader	Bootloader
Provisioning	x	NA	RSS	RSS
Provisioned, TZ_Closed, Closed, Locked	x	SECBOOTADD[31:8]	Boot address defined by user option byte SECBOOTADD[31:8]	Flash: 0x0C00 0000

Note: The BOOT_UBE is available only on STM32H533/73xx devices. Refer to Section 4.2.

When TrustZone is enabled (TZEN = 0xB4), the boot space must be in secure area. The SECBOOTADD0[24:0] option bytes are used to select the boot secure memory address. A unique boot entry option can be selected by setting the SECBOOT_LOCK option bit.

4.2 STM32H533/73xx boot modes

Table 25 and *Table 26* provide the detail of the boot mode products when the TrustZone is, respectively, disabled (TZEN = 0xC3) and enabled (TZEN = 0xB4).

Table 25. Boot mode when TrustZone is disabled (TZEN = 0xC3) - STM32H533/73xx

PRODUCT_STATE	BOOT0 pin	BOOT_UBE FLASH_OPTSR [29:22]	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	NA	NSBOOTADD[31:8]	Boot address defined by user option byte NSBOOTADD[31:8]	Flash: 0x0800 0000
	1	NA	NA	Bootloader	Bootloader
Provisioning	x	NA	NA	RSS	RSS
Provisioned, Closed, Locked	x	NA	NSBOOTADD[31:8]	Boot address defined by user option byte NSBOOTADD[31:8]	Flash: 0x0800 0000

Table 26. Boot mode when TrustZone is enabled (TZEN = 0xB4) - STM32H533/73xx

PRODUCT_STATE	BOOT0 pin	BOOT_UBE FLASH_OPTSR [29:22]	Boot address option-byte selection	Boot area	ST programmed default value
Open	0	x	SECBOOTADD [31:8]	Boot address defined by user option byte SECBOOTADD[31:8]	Flash: 0x0C00 0000
-	1	0xB4	NA	Bootloader	Bootloader
-	1	0xC3	NA	ST-iROT	ST-iROT
Provisioning	x	NA	NA	RSS	RSS
Provisioned, TZ_Closed, Closed, Locked	x	0xC3	ST-iROT	ST-iROT	ST-iROT
		0xB4	SECBOOTADD [31:8]	Boot address defined by user option byte SECBOOTADD[31:8]	Flash: 0x0C00 0000

When TrustZone is enabled (TZEN = 0xB4), the boot space must be in secure area. The SECBOOTADD0[24:0] option bytes are used to select the boot secure memory address. A unique boot entry option can be selected by setting the SECBOOT_LOCK option bit.

5 Global TrustZone® controller (GTZC)

5.1 GTZC introduction

The global TrustZone controller (GTZC) block that contains the following subblocks:

- **TZSC**: TrustZone security controller

  Defines the secure/privileged state of slave peripherals, and controls the subregion area size and properties for the watermark memory peripheral controller (MPCWM). The TZSC informs some peripherals (such as RCC or GPIOs) about the secure status of each securable peripheral, by sharing with RCC and I/O logic.
- **MPCBB**: memory protection controller - block based

  Configures the internal RAM in a TrustZone-system product having segmented SRAM (pages of 512 bytes) with programmable-security and privileged attributes.
- **TZIC**: TrustZone illegal access controller

  Gathers all illegal access events in the system, and generates secure interrupts towards NVIC.

These subblocks are used to configure TrustZone system security in a product having bus agents with programmable-security and privileged attributes such as:

- on-chip RAM with programmable secure and/or privileged blocks (pages)
- AHB and APB peripherals with programmable security and/or privileged access
- off-chip memories with secure and/or privileged areas

5.2 GTZC main features

- Three independent 32-bit AHB interfaces for TZSC, TZIC, and MPCBB
- TZIC accessible only with secure transactions
- Secure and non-secure access supported for privileged and unprivileged parts of TZSC and MPCBB
- Set of registers to define product security settings:
  - Secure/privileged blocks for internal SRAMs
  - Secure/privileged regions for external memories and internal backup SRAM
  - Secure/privileged access mode for securable peripherals
  - Secure/privileged access mode for securable masters

GTZC TrustZone system architecture

The Armv8-M supports security per TrustZone-M model with isolation between:

- a secure world, where usually security sensitive applications are run and critical resources are located
- a non-secure or public world (such as the usual non secure operating system and user space)

The TrustZone architecture is extended beyond AHB and Armv8-M with:

- AHB/APB bridge used as secure gate to block or propagate secure/non-secure and privileged/unprivileged transaction towards APB agents

- PPC (peripheral protection controller) used as secure gate to block or propagate secure/non-secure and privileged/unprivileged transaction towards AHB agents
- TrustZone block-based MPC firewalls used as secure gate to filter secure/non-secure, privileged/unprivileged access towards internal SRAMs
- TrustZone watermark MPC firewalls used as secure gate to filter secure/non-secure, privileged/unprivileged access towards external memories

AHB and APB peripherals can be categorized as:

- **privileged**: peripherals protected by an AHB/APB firewall stub controlled from TZSC to define privilege properties
- **secure**: peripherals always protected by an AHB/APB firewall stub, always secure (such as TZIC)
- **securable**: peripherals protected by an AHB/APB firewall stub that is controlled from TZSC to define security properties (optional)
- **non-secure and unprivileged:** peripherals connected directly to AHB/APB interconnect without any secure gate
- **TrustZone-aware**: peripherals connected directly to the AHB or APB bus and implementing a specific TrustZone behavior (such as a subset of registers being secure). TrustZone-aware AHB masters always drive HNONSEC signal according to their security mode (such as Armv8-M core or DMA)

AHB securable masters can be configured in the TZSC to be secure/non-secure and/or privileged/unprivileged.

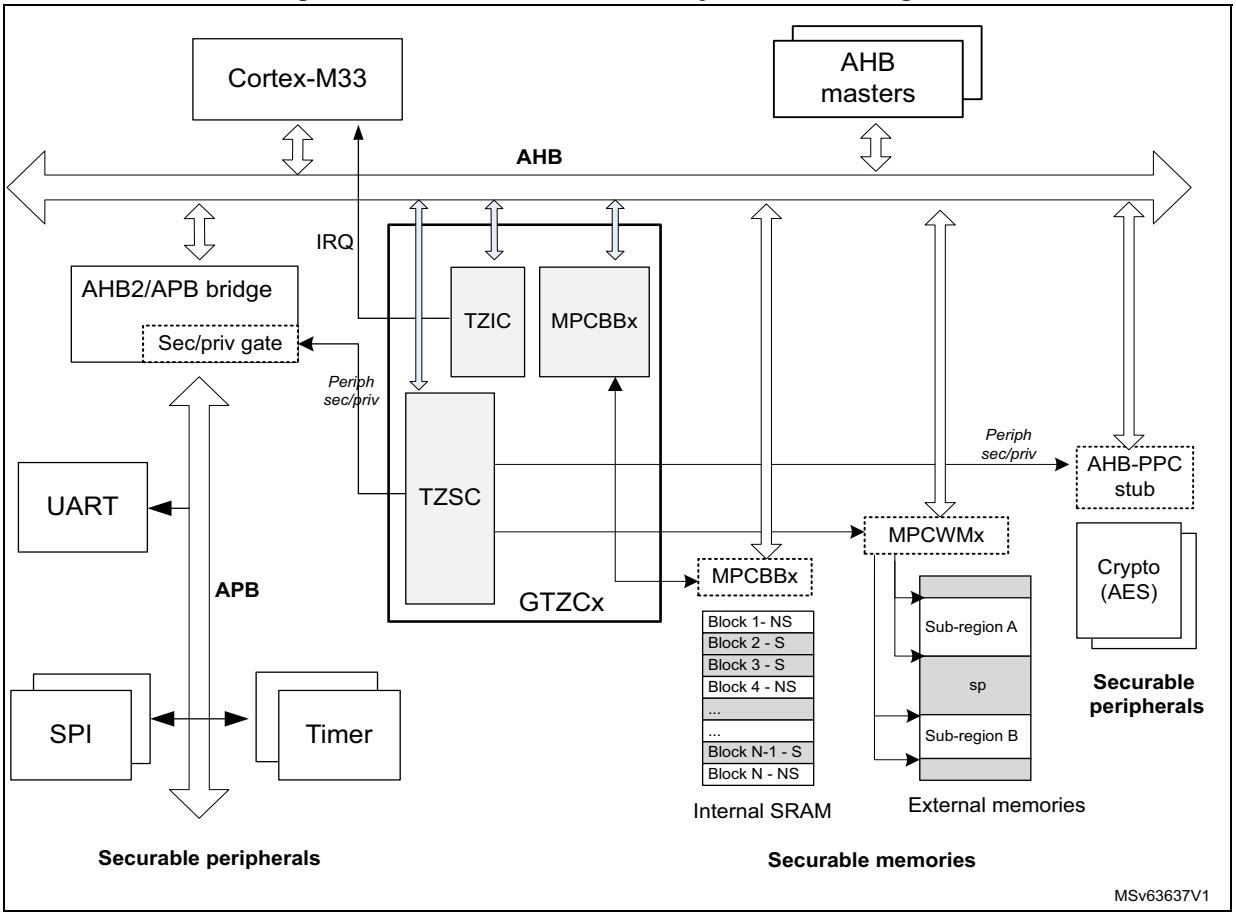
Application information

The TZSC, MPCBB, and TZIC can be used in one of the following ways:

- programmed only during secure boot, locked and not changed afterwards
- dynamically re-programmed when using specific application code or secure kernel (microvisor). When not locked, MPC secure blocks or region size can be changed by secure software executing from the secure FLASH region or secure SRAM. The same applies to the GTZC1_TZSC_SECCFGRx/PRIVCFGRx registers that define the secure/privileged state of each peripheral.

The Armv8-M security architecture with secure, securable, and TrustZone-aware peripherals is shown in *Figure 16*.

Figure 16. GTZC in Armv8-M subsystem block diagram



5.3 GTZC implementation

The STM32H5 devices embed one instance of GTZC.

Table 27. GTZC features

GTZC sub-blocks	GTZC1
TZSC	X
TZIC	X
MPCBB sub-block (number of MPCBBs)	X (3)

Table 28 shows the address offset of GTZC subblocks versus GTZC base address (refer to *Section 2.3* for GTZC1 base address).

Table 28. GTZC1 sub-block address offset

GTZC1 sub-block	Address offset
GTZC1_TZSC	0x0
GTZC1_TZIC	0x400
GTZC1_MPCBB1	0x800
GTZC1_MPCBB2	0xC00
GTZC1_MPCBB3	0x1000

Table 29 and *Table 30* describe the characteristics of the available MPCWMs.

Table 29. MPCWM resource assignment

GTZC	MPC	Target memory interface	Number of sec/non-sec and priv/unpriv regions	Watermark granularity (bytes)
GTZC1	MPCWM1	OCTOSPI1	2	128 K
	MPCWM2	FMC_NOR bank	2	128 K
	MPCWM3	FMC_NAND bank	1	128 K
		FMC_SDRAM bank 1(1)	1	128 K
	MPCWM4	BKPSRAM	1	32
		FMC_SDRAM_bank 2(1)	1	128 K

1. Not available on STM32H523/33xx devices.

Table 30. MPCWM3 and MPCWM4 (subregions A and B)

GTZ	MPC	Target memory interface	Subregion
GTZC1	MPCWM3	FMC_NAND bank	A
		FMC_SDRAM bank 1(1)	B
	MPCWM4	BKPSRAM	A
		FMC_SDRAM_bank 2(1)	B

1. Not available on STM32H523/33xx devices.

Table 31 and *Table 32* describe the characteristics of the available MPCBBs.

Table 31. MPCBB resource assignment (STM32H562/63/73xx devices)

GTZC	MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
GTZC1	MPCBB1	SRAM1	256	512	512	16
	MPCBB2	SRAM2	64		128	4
	MPCBB3	SRAM3	320		640	20

Table 32. MPCBB resource assignment (STM32H523/33xx devices)

GTZC	MPC	Resource	Memory size (Kbytes)	Block size (bytes)	Number of blocks	Number of super-blocks
GTZC1	MPCBB1	SRAM1	128	512	256	8
	MPCBB2	SRAM2	80		160	5
	MPCBB3	SRAM3	64		128	4

5.4 GTZC functional description

5.4.1 GTZC block diagram

Figure 17 describes the combined feature of TZSC, MPCBB, and TZIC. Each sub-block is controlled by its own AHB configuration port.

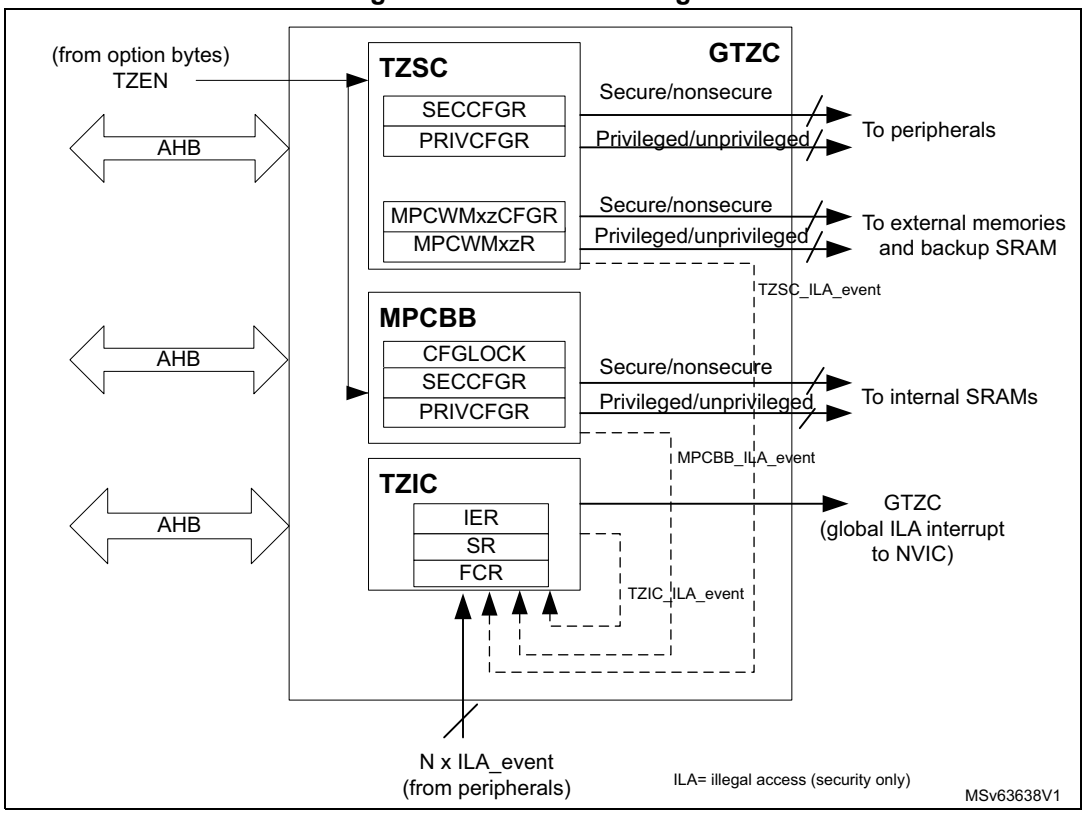
The TZSC defines which peripheral is secure and/or privileged. The privileged configuration bit of a peripheral can be modified by a secure privileged transaction when the peripheral is configured as secure. Otherwise, a privileged transaction (non-secure) is sufficient.

On the opposite, the secure configuration bit of a peripheral can be modified only with a secure privileged transaction if the peripheral is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The secure configuration bit of a given RAM block can be modified only with a secure privileged transaction if the same block is configured as privileged. Otherwise, a secure transaction (unprivileged) is sufficient.

The TZIC gathers illegal events generated within the system when an illegal access is detected. TZIC can then generate a secure interrupt towards the CPU if needed.

Figure 17. GTZC block diagram



5.4.2 Illegal access definition

Three different types of illegal access exist:

- Illegal non-secure access

  Any non-secure transaction trying to write a secure resource is considered as illegal, hence the addressed resource generates an illegal access interrupt for illegal write access and a bus error for illegal fetch access. There are some exceptions on secure and privileged configuration registers: the latter ones authorize non secure read access to secure registers (see GTZC1_TZSC_SECCFGRx/GTZC1_TZSC_PRIVCFGRx).

- Illegal secure access

  Any secure transaction trying to access non-secure block in internal block-based SRAM or watermarked memory is considered as illegal.

  A correct TZIC setting allows the capture of the associated event and then generates the GTZC_IRQn interrupt to the NVIC. This applies for read, write, and execute access.

  Concerning the MPCBB controller, there is an option to ignore secure data read/write access on non-secure SRAM blocks, by setting the SRWILADIS bit in the GTZC1_MPCBBz_CR register. Secure read and write data transactions are then

allowed on non-secure SRAM blocks, while secure execution access remains not allowed.

Any secure execute transaction trying to access a non-secure peripheral register is considered as illegal and generates a bus error.

- Illegal unprivileged access

  Any unprivileged transaction trying to access a privileged resource is considered as illegal. There is no illegal access event generated for illegal read and write accesses. The addressed resource follows a silent-fail behavior, returning all 0 data for read and ignoring any write. No bus error is generated. A bus error is generated when an unprivileged execute transaction tries to access a privileged memory.

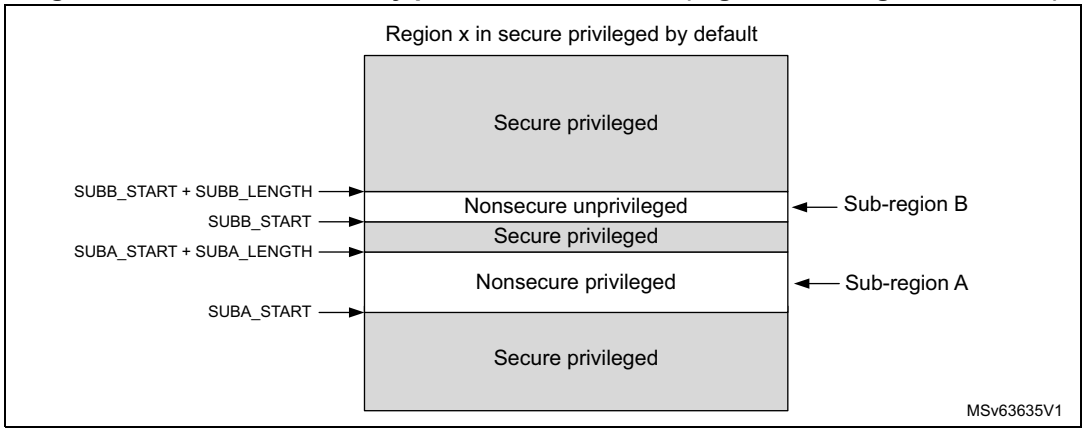
5.4.3 TrustZone security controller (TZSC)

The TZSC is composed of a configurable set of registers, providing the following features:

- Control of secure and privileged state for all peripherals, done through:
  - GTZC1_TZSC_SECCFGRx registers to control AHB/APB firewall stubs for the securable peripherals
  - GTZC1_TZSC_PRIVCFGRx registers to control AHB/APB firewall stubs for the privileged peripherals
- For watermark memory protection controller (external memories and backup SRAM), two independent regions can be defined and the following fields are used to program:
  - the start of the first protected subregion on the external memory/backup SRAM: SUBA_START[10:0]
  - the length of the first protected subregion on the external memory/backup SRAM: SUBA_LENGTH[11:0]
  - the start of the second protected subregion on the external memory/backup SRAM: SUBB_START[10:0]
  - the length of the second protected subregion on the external memory/backup SRAM: SUBB_LENGTH[11:0]

A control register for each subregion can be used to enable/disable the watermark memory protection controller, and to define the right attributes of each subregion.

Figure 18. Watermark memory protection controller (region x/subregions A and B)



In *Figure 18*, region x represents the external memory or backup SRAM region (such as FMC bank, OCTOSPI1, or BKPSRAM). Secure and privileged attributes of subregions A

and B are independently configurable. When no subregions are defined or enabled on the region x, the default attribute of the region x is set as secure-privileged.

The following tables describe the secure/privileged properties of the common area of subregions A and B, when an overlap exists.

Table 33. Secure properties of subregions A and B

Subregion A	Subregion B	Properties of overlapped region A and B
Non-secure	Non-secure	Non-secure
Non-secure	Secure	Non-secure
Secure	Non-secure	Non-secure
Secure	Secure	Secure

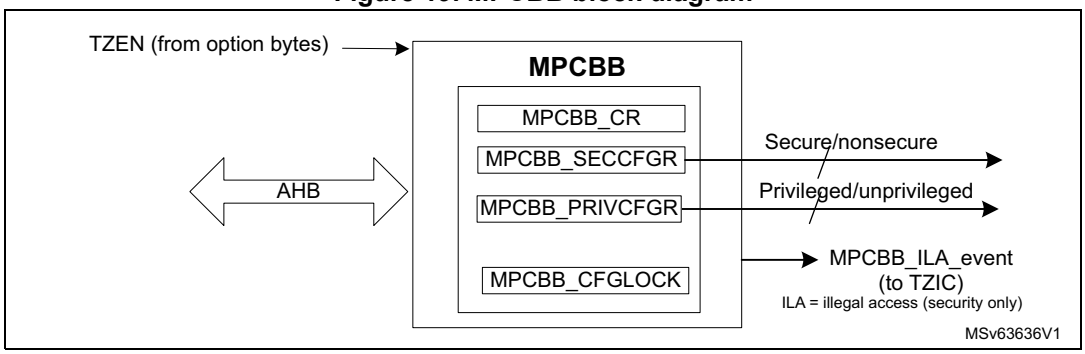
Table 34. Privileged properties of subregions A and B

Subregion A	Subregion B	Properties of overlapped region A and B
Unprivileged	Unprivileged	Unprivileged
Unprivileged	Privileged	Unprivileged
Privileged	Unprivileged	Unprivileged
Privileged	Privileged	Privileged

5.4.4 Memory protection controller - block based (MPCBB)

The MPCBB is composed of a configurable set of registers allowing to define security and privileged policy for internal SRAMs. The security and privileged policy can be individually configured per each 512-byte block.

Figure 19. MPCBB block diagram



To set up the MPCBB, the following actions are needed (for example at boot time):

- Secure firmware must define which memory blocks are secure by setting the correct bits in GTZC1_MPCBBz_SECCFGRx.
- Privileged firmware must define which memory blocks are privileged by setting the correct bits in GTZC1_MPCBBz_PRIVCFGRx.

An MPCBB super-block is made of 32 consecutive blocks. For each super-block, secure application can lock all related security/privileged bits using the correct bits in GTZC1_MPCBBz_CFGLOCK. This lock remains active until the next system reset.

*Note: The block size is 512 bytes. The super-block size is 512 * 32 = 16 Kbytes.*

5.4.5 TrustZone illegal access controller (TZIC)

The TZIC concentrates all illegal access source events. It is used only when the system is TrustZone enabled (TZEN = 0xB4).

TZIC allows to trace (flag) the event that triggered the secure illegal access interrupt. Register masks (GTZC1_TZIC_IERx) are available to filter unwanted event. On an unmasked illegal event, TZIC generates the GTZC_IRQn interrupt to the NVIC.

For each illegal event source, a status flag and a clear bit exist (respectively within GTZC1_TZIC_SRx and GTZC1_TZIC_FCRx). The reset value of mask registers (GTZC1_TZIC_IERx) is such that all events are masked.

5.4.6 Power-on/reset state

The power-on and reset state of the TZSC clear to 0 all bits of GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx, meaning that all securable peripherals are respectively set to non-secure and unprivileged.

For internal SRAMx (x = 1 to 3), all GTZC1_MPCBBz_SECCFGRx and GTZC1_MPCBBz_PRIVCFGRx are set:

- to 0xFFFF FFFF, making these internal memories block secure and privileged by default when TrustZone security is enabled at system level (TZEN = 0xB4)
- to 0x0000 0000, making these internal memories block non-secure and unprivileged by default when TrustZone security is disabled at system level (TZEN = 0xC3)

For external memories and backup SRAM, GTZC1_TZSC_MPCWMxzR registers are set:

- to 0x0000 0000, making these memories secure and privileged by default when TrustZone security is enabled a system level (TZEN = 0xB4).
- to 0x0800 0000, making these memories non-secure and non-privileged by default when TrustZone security is disabled at system level (TZEN = 0xC3)

Secure boot code can then program the security settings, making components secure or not as needed.

5.5 GTZC interrupts

TZIC is a secure peripheral, thus it systematically generates an illegal access event when accessed by a non-secure access. The MPCBB and TZSC are TrustZone-aware peripherals, meaning that secure and non-secure registers coexist within the peripheral.

Table 35. GTZC interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
GTZC	Illegal access	All flags in GTZC1_TZIC_SRx	All bits in GTZC1_TZIC_IERx	Write 1 in the bit GTZC1_TZIC_FCRx	Yes	Yes	No

5.6 GTZC1 TZSC registers

All registers are accessed only by words (32-bit).

5.6.1 GTZC1 TZSC control register (GTZC1_TZSC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

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

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

Bit 0 **LCK**: lock the configuration of GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx until next reset

This bit is cleared by default, and once set, it cannot be reset until system reset.

0: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx not locked

1: configuration of all GTZC1_TZSC_SECCFGRx and GTZC1_TZSC_PRIVCFGRx locked

5.6.2 GTZC1 TZSC secure configuration register 1 (GTZC1_TZSC_SECCFGR1)

Address offset: 0x010

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR signal is set to 1. If a given PRIV bit is not set, the equivalent SEC bit can be written by secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM2SEC	DTSSEC	UART12SEC	UART9SEC	UART8SEC	UART7SEC	DAC1SEC	HDMICECSEC	USART11SEC	USART10SEC	USART6SEC	CRSSEC	I3C1SEC	I2C2SEC	I2C1SEC	UART5SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4SEC	USART3SEC	USART2SEC	SPI3SEC	SPI2SEC	IWDGSEC	WWDGSEC	TIM14SEC	TIM13SEC	TIM12SEC	TIM7SEC	TIM6SEC	TIM5SEC	TIM4SEC	TIM3SEC	TIM2SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM2SEC**: secure access mode for LPTIM2
0: non-secure
1: secure

Bit 30 **DTSSEC**: secure access mode for DTS
0: non-secure
1: secure

Bit 29 **UART12SEC**: secure access mode for UART12
0: non-secure
1: secure

Bit 28 **UART9SEC**: secure access mode for UART9
0: non-secure
1: secure

Bit 27 **UART8SEC**: secure access mode for UART8
0: non-secure
1: secure

Bit 26 **UART7SEC**: secure access mode for UART7
0: non-secure
1: secure

Bit 25 **DAC1SEC**: secure access mode for DAC1
0: non-secure
1: secure

Bit 24 **HDMICECSEC**: secure access mode for HDMICEC
0: non-secure
1: secure

Bit 23 **USART11SEC**: secure access mode for USART11
0: non-secure
1: secure

Bit 22 **USART10SEC**: secure access mode for USART10
0: non-secure
1: secure

Bit 21 **USART6SEC**: secure access mode for USART6
0: non-secure
1: secure

Bit 20 **CRSSEC**: secure access mode for CRS
0: non-secure
1: secure

Bit 19 **I3C1SEC**: secure access mode for I3C1
0: non-secure
1: secure

Bit 18 **I2C2SEC**: secure access mode for I2C2
0: non-secure
1: secure

Bit 17 **I2C1SEC**: secure access mode for I2C1
0: non-secure
1: secure

Bit 16 **UART5SEC**: secure access mode for UART5
0: non-secure
1: secure

Bit 15 **UART4SEC**: secure access mode for UART4
0: non-secure
1: secure

Bit 14 **USART3SEC**: secure access mode for USART3
0: non-secure
1: secure

Bit 13 **USART2SEC**: secure access mode for USART2
0: non-secure
1: secure

Bit 12 **SPI3SEC**: secure access mode for SPI3
0: non-secure
1: secure

Bit 11 **SPI2SEC**: secure access mode for SPI2
0: non-secure
1: secure

Bit 10 **IWDGSEC**: secure access mode for IWDG
0: non-secure
1: secure

Bit 9 **WWDGSEC**: secure access mode for WWDG
0: non-secure
1: secure

Bit 8 **TIM14SEC**: secure access mode for TIM14
0: non-secure
1: secure

Bit 7 **TIM13SEC**: secure access mode for TIM13
0: non-secure
1: secure

Bit 6 **TIM12SEC**: secure access mode for TIM12
0: non-secure
1: secure

Bit 5 **TIM7SEC**: secure access mode for TIM7
0: non-secure
1: secure

Bit 4 **TIM6SEC**: secure access mode for TIM6
0: non-secure
1: secure

Bit 3 **TIM5SEC**: secure access mode for TIM5
0: non-secure
1: secure

Bit 2 **TIM4SEC**: secure access mode for TIM4
0: non-secure
1: secure

Bit 1 **TIM3SEC**: secure access mode for TIM3
0: non-secure
1: secure

Bit 0 **TIM2SEC**: secure access mode for TIM2
0: non-secure
1: secure

5.6.3 GTZC1 TZSC secure configuration register 2 (GTZC1_TZSC_SECCFGR2)

Address offset: 0x014

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR signal is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by a secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM5 SEC	LPTIM4 SEC	LPTIM3 SEC	LPTIM1 SEC	I2C4 SEC	I2C3 SEC	LPUART1 SEC	SPI5 SEC	Res.	Res.	Res.	Res.	USB SEC	SAI2 SEC	SAI1 SEC	SPI6 SEC
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI4 SEC	TIM17 SEC	TIM16 SEC	TIM15 SEC	USART 1SEC	TIM8 SEC	SPI1 SEC	TIM1 SEC	Res.	Res.	Res.	Res.	Res.	UCPD SEC	FDCAN2 SEC	FDCAN1 SEC
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bit 31 **LPTIM5SEC**: secure access mode for LPTIM5
0: non-secure
1: secure

Bit 30 **LPTIM4SEC**: secure access mode for LPTIM4
0: non-secure
1: secure

Bit 29 **LPTIM3SEC**: secure access mode for LPTIM3
0: non-secure
1: secure

Bit 28 **LPTIM1SEC**: secure access mode for LPTIM1
0: non-secure
1: secure

Bit 27 **I2C4SEC**: secure access mode for I2C4
0: non-secure
1: secure

Bit 26 **I2C3SEC**: secure access mode for I2C3
0: non-secure
1: secure

Bit 25 **LPUART1SEC**: secure access mode for LPUART
0: non-secure
1: secure

Bit 24 **SPI5SEC**: secure access mode for SPI5
0: non-secure
1: secure

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **USBSEC**: secure access mode for USB
0: non-secure
1: secure

Bit 18 **SAI2SEC**: secure access mode for SAI2
0: non-secure
1: secure

Bit 17 **SAI1SEC**: secure access mode for SAI1
0: non-secure
1: secure

Bit 16 **SPI6SEC**: secure access mode for SPI6
0: non-secure
1: secure

Bit 15 **SPI4SEC**: secure access mode for SPI4
0: non-secure
1: secure

Bit 14 **TIM17SEC**: secure access mode for TIM17
0: non-secure
1: secure

Bit 13 **TIM16SEC**: secure access mode for TIM16
0: non-secure
1: secure

Bit 12 **TIM15SEC**: secure access mode for TIM15
0: non-secure
1: secure

Bit 11 **USART1SEC**: secure access mode for USART1
0: non-secure
1: secure

Bit 10 **TIM8SEC**: secure access mode for TIM8
0: non-secure
1: secure

Bit 9 **SPI1SEC**: secure access mode for SPI1
0: non-secure
1: secure

Bit 8 **TIM1SEC**: secure access mode for TIM1
0: non-secure
1: secure

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **UCPDSEC**: secure access mode for UCPD
0: non-secure
1: secure

Bit 1 **FDCAN2SEC**: secure access mode for FDCAN2
0: non-secure
1: secure

Bit 0 **FDCAN1SEC**: secure access mode for FDCAN1
0: non-secure
1: secure

5.6.4 GTZC1 TZSC secure configuration register 3 (GTZC1_TZSC_SECCFGR3)

Address offset: 0x018

Reset value: 0x0000 0000

Write-secure access only.

This register can be written only by a secure privileged transaction when the corresponding GTZC1_TZSC_PRIVCFGR is set to 1. If a given PRIV is not set, the equivalent SEC bit can be written by a secure unprivileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RAMCFGSEC	Res.	OCTOSPI1SEC	FMCSEC	SDMMC2SEC	SDMMC1SEC	PKASEC	SAESSEC	RNGSEC	HASHSEC	AESSEC
					rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCMISEC	ADC12SEC	DCACHESEC	ICACHESEC	ETHSEC	FMACSEC	CORDICSEC	CRCSEC	Res.	Res.	Res.	Res.	Res.	I3C2SEC	VREFBUFSEC	LPTIM6SEC
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RAMCFGSEC**: secure access mode for RAMSCFG
0: non-secure
1: secure

Bit 25 Reserved, must be kept at reset value.

Bit 24 **OCTOSPI1SEC**: secure access mode for OCTOSPI1
0: non-secure
1: secure

Bit 23 **FMCSEC**: secure access mode for FMC
0: non-secure
1: secure

Bit 22 **SDMMC2SEC**: secure access mode for SDMMC2
0: non-secure
1: secure

Bit 21 **SDMMC1SEC**: secure access mode for SDMMC1
0: non-secure
1: secure

Bit 20 **PKASEC**: secure access mode for PKA
0: non-secure
1: secure

Bit 19 **SAESSEC**: secure access mode for SAES
0: non-secure
1: secure

Bit 18 **RNGSEC**: secure access mode for RNG
0: non-secure
1: secure

Bit 17 **HASHSEC**: secure access mode for HASH
0: non-secure
1: secure

Bit 16 **AESSEC**: secure access mode for AES
0: non-secure
1: secure

Bit 15 **DCMISEC**: secure access mode for DCMI
0: non-secure
1: secure

Bit 14 **ADC12SEC**: secure access mode for ADC1 and ADC2
0: non-secure
1: secure

Bit 13 **DCACHESEC**: secure access mode for DCACHE
0: non-secure
1: secure

Bit 12 **ICACHESEC**: secure access mode for ICACHE
0: non-secure
1: secure

Bit 11 **ETHSEC**: secure access mode for register of ETH
0: non-secure
1: secure

Bit 10 **FMACSEC**: secure access mode for FMAC
0: non-secure
1: secure

Bit 9 **CORDICSEC**: secure access mode for CORDIC
0: non-secure
1: secure

Bit 8 **CRCSEC**: secure access mode for CRC
0: non-secure
1: secure

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **I3C2SEC**: secure access mode for I3C2
0: non-secure
1: secure

Bit 1 **VREFBUFSEC**: secure access mode for VREFBUF
0: non-secure
1: secure

Bit 0 **LPTIM6SEC**: secure access mode for LPTIM6
0: non-secure
1: secure

5.6.5 GTZC1 TZSC privilege configuration register 1 (GTZC1_TZSC_PRIVCFGR1)

Address offset: 0x020

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by a secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by a non-secure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM2PRIV	DTSPRIV	UART12PRIV	UART9PRIV	UART8PRIV	UART7PRIV	DAC1PRIV	HDMICECPRIV	USART11PRIV	USART10PRIV	USART6PRIV	CRSPRIV	I3C1PRIV	I2C2PRIV	I2C1PRIV	UART5PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4PRIV	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM14PRIV	TIM13PRIV	TIM12PRIV	TIM7PRIV	TIM6PRIV	TIM5PRIV	TIM4PRIV	TIM3PRIV	TIM2PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM2PRIV**: privileged access mode for LPTIM2
0: unprivileged
1: privileged

Bit 30 **DTSPRIV**: privileged access mode for DTS
0: unprivileged
1: privileged

Bit 29 **UART12PRIV**: privileged access mode for UART12
0: unprivileged
1: privileged

Bit 28 **UART9PRIV**: privileged access mode for UART9
0: unprivileged
1: privileged

Bit 27 **UART8PRIV**: privileged access mode for UART8
0: unprivileged
1: privileged

Bit 26 **UART7PRIV**: privileged access mode for UART7
0: unprivileged
1: privileged

Bit 25 **DAC1PRIV**: privileged access mode for DAC1
0: unprivileged
1: privileged

Bit 24 **HDMICECPRIV**: privileged access mode for HDMICEC
0: unprivileged
1: privileged

Bit 23 **USART11PRIV**: privileged access mode for USART11
0: unprivileged
1: privileged

Bit 22 **USART10PRIV**: privileged access mode for USART10
0: unprivileged
1: privileged

Bit 21 **USART6PRIV**: privileged access mode for USART6
0: unprivileged
1: privileged

Bit 20 **CRSPRIV**: privileged access mode for CRS
0: unprivileged
1: privileged

Bit 19 **I3C1PRIV**: privileged access mode for I3C1
0: unprivileged
1: privileged

Bit 18 **I2C2PRIV**: privileged access mode for I2C2
0: unprivileged
1: privileged

Bit 17 **I2C1PRIV**: privileged access mode for I2C1
0: unprivileged
1: privileged

Bit 16 **UART5PRIV**: privileged access mode for UART5
0: unprivileged
1: privileged

Bit 15 **UART4PRIV**: privileged access mode for UART4
0: unprivileged
1: privileged

Bit 14 **USART3PRIV**: privileged access mode for USART3
0: unprivileged
1: privileged

Bit 13 **USART2PRIV**: privileged access mode for USART2
0: unprivileged
1: privileged

Bit 12 **SPI3PRIV**: privileged access mode for SPI3
0: unprivileged
1: privileged

Bit 11 **SPI2PRIV**: privileged access mode for SPI2
0: unprivileged
1: privileged

Bit 10 **IWDGPRIV**: privileged access mode for IWDG
0: unprivileged
1: privileged

Bit 9 **WWDGPRIV**: privileged access mode for WWDG
0: unprivileged
1: privileged

Bit 8 **TIM14PRIV**: privileged access mode for TIM14
0: unprivileged
1: privileged

Bit 7 **TIM13PRIV**: privileged access mode for TIM13
0: unprivileged
1: privileged

Bit 6 **TIM12PRIV**: privileged access mode for TIM12
0: unprivileged
1: privileged

Bit 5 **TIM7PRIV**: privileged access mode for TIM7
0: unprivileged
1: privileged

Bit 4 **TIM6PRIV**: privileged access mode for TIM6
0: unprivileged
1: privileged

Bit 3 **TIM5PRIV**: privileged access mode for TIM5
0: unprivileged
1: privileged

Bit 2 **TIM4PRIV**: privileged access mode for TIM4
0: unprivileged
1: privileged

Bit 1 **TIM3PRIV**: privileged access mode for TIM3
0: unprivileged
1: privileged

Bit 0 **TIM2PRIV**: privileged access mode for TIM2
0: unprivileged
1: privileged

5.6.6 GTZC1 TZSC privilege configuration register 2 (GTZC1_TZSC_PRIVCFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by a secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by a non-secure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM5 PRIV	LPTIM4 PRIV	LPTIM3 PRIV	LPTIM1 PRIV	I2C4PR IV	I2C3PR IV	LPUAR T1PRIV	SPI5P RIV	Res.	Res.	Res.	Res.	USBPR IV	SAI2P RIV	SAI1P RIV	SPI6P RIV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI4P RIV	TIM17P RIV	TIM16P RIV	TIM15P RIV	USART 1PRIV	TIM8P RIV	SPI1P RIV	TIM1P RIV	Res.	Res.	Res.	Res.	Res.	UCPDP RIV	FDCAN 2PRIV	FDCAN 1PRIV
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bit 31 **LPTIM5PRIV**: privileged access mode for LPTIM5
0: unprivileged
1: privileged

Bit 30 **LPTIM4PRIV**: privileged access mode for LPTIM4
0: unprivileged
1: privileged

Bit 29 **LPTIM3PRIV**: privileged access mode for LPTIM3
0: unprivileged
1: privileged

Bit 28 **LPTIM1PRIV**: privileged access mode for LPTIM1
0: unprivileged
1: privileged

Bit 27 **I2C4PRIV**: privileged access mode for I2C4
0: unprivileged
1: privileged

Bit 26 **I2C3PRIV**: privileged access mode for I2C3
0: unprivileged
1: privileged

Bit 25 **LPUART1PRIV**: privileged access mode for LPUART
0: unprivileged
1: privileged

Bit 24 **SPI5PRIV**: privileged access mode for SPI5
0: unprivileged
1: privileged

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **USBPRIV**: privileged access mode for USB
0: unprivileged
1: privileged

Bit 18 **SAI2PRIV**: privileged access mode for SAI2
0: unprivileged
1: privileged

Bit 17 **SAI1PRIV**: privileged access mode for SAI1
0: unprivileged
1: privileged

Bit 16 **SPI6PRIV**: privileged access mode for SPI6
0: unprivileged
1: privileged

Bit 15 **SPI4PRIV**: privileged access mode for SPI4
0: unprivileged
1: privileged

Bit 14 **TIM17PRIV**: privileged access mode for TIM17
0: unprivileged
1: privileged

Bit 13 **TIM16PRIV**: privileged access mode for TIM16
0: unprivileged
1: privileged

Bit 12 **TIM15PRIV**: privileged access mode for TIM15
0: unprivileged
1: privileged

Bit 11 **USART1PRIV**: privileged access mode for USART1
0: unprivileged
1: privileged

Bit 10 **TIM8PRIV**: privileged access mode for TIM8
0: unprivileged
1: privileged

Bit 9 **SPI1PRIV**: privileged access mode for SPI1
0: unprivileged
1: privileged

Bit 8 **TIM1PRIV**: privileged access mode for TIM1
0: unprivileged
1: privileged

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **UCPDPRIV**: privileged access mode for UCPD
0: unprivileged
1: privileged

Bit 1 **FDCAN2PRIV**: privileged access mode for FDCAN2
0: unprivileged
1: privileged

Bit 0 **FDCAN1PRIV**: privileged access mode for FDCAN1
0: unprivileged
1: privileged

5.6.7 GTZC1 TZSC privilege configuration register 3 (GTZC1_TZSC_PRIVCFGR3)

Address offset: 0x028

Reset value: 0x0000 0000

Write-privileged access only.

This register can be read or written only by a secure privileged transaction when the corresponding GTZC1_TZSC_SECCFGR signal is set to1. If a given SEC bit is not set, the equivalent PRIV bit can be read/written by a non-secure privileged transaction.

Read accesses are authorized for any type of transactions, secure or not, privileged or not.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RAMCFG PRIV	Res.	OCTOSPI1 PRIV	FMC PRIV	SDMMC2 PRIV	SDMMC1 PRIV	PKA PRIV	SAES PRIV	RNG PRIV	HASH PRIV	AES PRIV
					rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCMI PRIV	ADC12 PRIV	DCACHE PRIV	ICACHE PRIV	ETH PRIV	FMAC PRIV	CORDIC PRIV	CRC PRIV	Res.	Res.	Res.	Res.	Res.	I3C2 PRIV	VREFBUF PRIV	LPTIM6 PRIV
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RAMCFGPRIV**: privileged access mode for RAMSCFG
0: unprivileged
1: privileged

Bit 25 Reserved, must be kept at reset value.

Bit 24 **OCTOSPI1PRIV**: privileged access mode for OCTOSPI1
0: unprivileged
1: privileged

Bit 23 **FMCPRIV**: privileged access mode for FMC
0: unprivileged
1: privileged

Bit 22 **SDMMC2PRIV**: privileged access mode for SDMMC2
0: unprivileged
1: privileged

Bit 21 **SDMMC1PRIV**: privileged access mode for SDMMC1
0: unprivileged
1: privileged

Bit 20 **PKAPRIV**: privileged access mode for PKA
0: unprivileged
1: privileged

Bit 19 **SAESPRIV**: privileged access mode for SAES
0: unprivileged
1: privileged

Bit 18 **RNGPRIV**: privileged access mode for RNG
0: unprivileged
1: privileged

Bit 17 **HASHPRIV**: privileged access mode for HASH
0: unprivileged
1: privileged

Bit 16 **AESPRIV**: privileged access mode for AES
0: unprivileged
1: privileged

Bit 15 **DCMIPRIV**: privileged access mode for DCMI
0: unprivileged
1: privileged

Bit 14 **ADC12PRIV**: privileged access mode for ADC1 and ADC2
0: unprivileged
1: privileged

Bit 13 **DCACHEPRIV**: privileged access mode for DCACHE
0: unprivileged
1: privileged

Bit 12 **ICACHEPRIV**: privileged access mode for ICACHE
0: unprivileged
1: privileged

Bit 11 **ETHPRIV**: privileged access mode for register of ETH
0: unprivileged
1: privileged

Bit 10 **FMACPRIV**: privileged access mode for FMAC
0: unprivileged
1: privileged

Bit 9 **CORDICPRIV**: privileged access mode for CORDIC
0: unprivileged
1: privileged

Bit 8 **CRCPRIV**: privileged access mode for CRC
0: unprivileged
1: privileged

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **I3C2PRIV**: privileged access mode for I3C2
0: unprivileged
1: privileged

Bit 1 **VREFBUFPRIV**: privileged access mode for VREFBUF
0: unprivileged
1: privileged

Bit 0 **LPTIM6PRIV**: privileged access mode for LPTIM6
0: unprivileged
1: privileged

5.6.8 GTZC1 TZSC memory x subregion A watermark configuration register (GTZC1_TZSC_MPCWMxACFGR)

Address offset: 0x40 + 0x10 *(x - 1) (x = 1 to 4)

Reset value: 0x0000 0000

Secure privilege access only.

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

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

Bit 9 **PRIV**: Privileged subregion A of base region x
This bit is taken into account only if SREN is set.
0: Privileged and unprivileged accesses are granted in subregion A.
1: Only privileged accesses are granted in subregion A of region x.

Bit 8 **SEC**: Secure subregion A of base region x
This bit is taken into account only if SREN is set.
0: Only non-secure data accesses are granted to subregion A of region x.
1: Only secure data accesses are granted to subregion A of region x.

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

Bit 1 **SRLOCK**: Subregion A lock
This bit, once set, can be cleared only by a system reset.
0: GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR can be written.
1: Writes to GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR are ignored.

Bit 0 **SREN**: Subregion A enable
0: subregion A is disabled. Access control of base region x applies to any access between this subregion start- and end-addresses.
1: subregion A of region x is enabled. Access control defined in GTZC1_TZSC_MPCWMx_CFGR applies to any access between this subregion start- and end-addresses, defined in GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR.

Note: Watermark controlled external memories start fully non-secure/unprivileged at reset when TZEN = 0xC3. When TZEN = 0xB4, they start fully secure/fully privileged (inverted reset-value).

5.6.9 GTZC1 TZSC memory x subregion A watermark register (GTZC1_TZSC_MPCWMxAR)

Address offset: 0x44 + 0x10 *(x - 1) (x = 1 to 4)

Reset value: 0x0000 0000

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0800 0000 when TZEN = 0xC3.

Secure privilege access only.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both subregions apply on the common section (see *Section 5.4.3*)

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SUBA_LENGTH[11:0]**: Length of subregion A in region x

This field defines the length of the subregion A, to be multiplied by the granularity defined in *Table 29*.

When SUBA_START + SUBA_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBA_LENGTH is applied automatically. If SUBA_LENGTH = 0, subregion A is disabled (SREN bit in GTZC1_TZSC_MPCMWxACFGR is cleared).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **SUBA_START[10:0]**: Start of subregion A in region x

This field defines the address offset of the subregion A, to be multiplied by the granularity defined in *Table 29*, versus the start of the region x.

Watermark controlled external memories start fully non-secure at reset when TZEN = 0xC3. When TZEN = 0xB4, they start fully secure (inverted reset value).

5.6.10 GTZC1 TZSC memory x subregion B watermark configuration register (GTZC1_TZSC_MPCWMxBCFGR)

Address offset: 0x48 + 0x10 *(x - 1) (x = 1 to 4)

Reset value: 0x0000 0000

Secure privilege access only.

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

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

Bit 9 **PRIV**: Privileged subregion B of base region x

This bit is taken into account only if SREN is set.

0: Privileged and unprivileged accesses are granted in subregion A.

1: Only privileged accesses are granted in subregion A of region x.

Bit 8 **SEC**: Secure subregion B of base region x

This bit is taken into account only if SREN is set.

0: Only non-secure data accesses are granted to subregion A of region x.

1: Only secure data accesses are granted to subregion A of region x.

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

Bit 1 **SRLOCK**: Subregion B lock

This bit, once set, can be cleared only by a system reset.

0: GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR can be written.

1: Writes to GTZC1_TZSC_MPCWMxCFGR, GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR are ignored.

Bit 0 **SREN**: Subregion B enable

0: subregion B is disabled. Access control of base region x applies to any access between this subregion start- and end-addresses.

1: subregion B of region x is enabled. Access control defined in GTZC1_TZSC_MPCWMx_CFGR applies to any access between this subregion start- and end-addresses, defined in GTZC1_TZSC_MPCWMxAR and GTZC1_TZSC_MPCWMxBR.

Note: Watermark controlled external memories start fully non-secure/unprivileged at reset when TZEN = 0xC3. When TZEN = 0xB4, they start fully secure/fully privileged (inverted reset-value).

5.6.11 GTZC1 TZSC memory x subregion B watermark register (GTZC1_TZSC_MPCWMxBR)

Address offset: 0x4C + 0x10 *(x - 1) (x = 1 to 4)

Reset value: 0x0000 0000

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0800 0000 when TZEN = 0xC3.

Secure privilege access only.

When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.

When an overlap of subregion A and B exists, secure/privileged attributes of both subregions apply on the common section (see *Section 5.4.3*)

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **SUBB_LENGTH[11:0]**: Length of subregion B in region x

This field defines the length of the subregion B, to be multiplied by the granularity defined in *Table 29*.

When SUBB_START + SUBB_LENGTH is higher than the maximum size allowed for the memory, a saturation of SUBB_LENGTH is applied automatically.

If SUBB_LENGTH = 0, the subregion B is disabled (SREN bit in GTZC1_TZSC_MPCMWxBCFGR is cleared).

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **SUBB_START[10:0]**: Start of subregion B in region x

This field defines the address offset of the subregion B, to be multiplied by the granularity defined in *Table 29*, versus the start of the region x.

Wtermark controlled external memories start fully non-secure at reset when TZEN = 0xC3. When TZEN = 0xB4, they start fully secure (inverted reset value).

5.6.12 GTZC1 TZSC register map

Table 36. GTZC1 TZSC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_TZSC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCK
	Reset value																																0
0x004-0x00C	Reserved	Reserved																															
0x010	GTZC1_TZSC_SECCFGR1	LPTIM2SEC	DTSSEC	UART12SEC	UART9SEC	UART8SEC	UART7SEC	DAC1SEC	HDMICECSEC	USART11SEC	USART10SEC	USART6SEC	CRSSEC	I3C1SEC	I2C2SEC	I2C1SEC	UART5SEC	UART4SEC	USART3SEC	USART2SEC	SPI3SEC	SPI2SEC	IWDGSEC	WWDGSEC	TIM14SEC	TIM13SEC	TIM112SEC	TIM7SEC	TIM6SEC	TIM5SEC	TIM4SEC	TIM3SEC	TIM2SEC
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	GTZC1_TZSC_SECCFGR2	LPTIM5SEC	LPTIM4SEC	LPTIM3SEC	LPTIM1SEC	I2C4SEC	I2C3SEC	LPUART1SEC	SPI5SEC	Res.	Res.	Res.	Res.	USBSEC	SAI2SEC	SAI1SEC	SPI6SEC	SPI4SEC	TIM17SEC	TIM16SEC	TIM15SEC	USART1SEC	TIM8SEC	SPI1SEC	TIM1SEC	Res.	Res.	Res.	Res.	Res.	UCPDSEC	FDCAN2SEC	FDCAN1SEC
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x018	GTZC1_TZSC_SECCFGR3	Res.	Res.	Res.	Res.	Res.	RAMCFGSEC	Res.	OCTOSPI1SEC	FMCSEC	SDMMC2SEC	SDMMC1SEC	PKASEC	SAESSEC	RNGSEC	HASHSEC	AESSEC	DCMISEC	ADC12SEC	DCACHESEC	ICACHESEC	ETHSEC	FMACSEC	CORDICSEC	CRCSEC	Res.	Res.	Res.	Res.	Res.	I3C2SEC	VREFBUFSEC	LPTIM6SEC
	Reset value						0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x01C	Reserved	Reserved																															
0x020	GTZC1_TZSC_PRIVCFGR1	LPTIM2PRIV	DTSPRIV	UART12PRIV	UART9PRIV	UART8PRIV	UART7PRIV	DAC1PRIV	HDMICECPRIV	USART11PRIV	USART10PRIV	USART6PRIV	CRSPRIV	I3C1PRIV	I2C2PRIV	I2C1PRIV	UART5PRIV	UART4PRIV	USART3PRIV	USART2PRIV	SPI3PRIV	SPI2PRIV	IWDGPRIV	WWDGPRIV	TIM14PRIV	TIM13PRIV	TIM112PRIV	TIM7PRIV	TIM6PRIV	TIM5PRIV	TIM4PRIV	TIM3PRIV	TIM2PRIV
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	GTZC1_TZSC_PRIVCFGR2	LPTIM5PRIV	LPTIM4PRIV	LPTIM3PRIV	LPTIM1PRIV	I2C4PRIV	I2C3PRIV	LPUART1PRIV	SPI5PRIV	Res.	Res.	Res.	Res.	USBPRIV	SAI2PRIV	SAI1PRIV	SPI6PRIV	SPI4PRIV	TIM17PRIV	TIM16PRIV	TIM15PRIV	USART1PRIV	TIM8PRIV	SPI1PRIV	TIM1PRIV	Res.	Res.	Res.	Res.	Res.	UCPDPRIV	FDCAN2PRIV	FDCAN1PRIV
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x028	GTZC1_TZSC_PRIVFGR3	Res.	Res.	Res.	Res.	Res.	RAMCFGPRIV	Res.	OCTOSPI1PRIV	FMCPRIV	SDMMC2PRIV	SDMMC1PRIV	PKAPRIV	SAESPRIV	RNGPRIV	HASHPRIV	AESPRIV	DCMIPRIV	ADC12PRIV	DCACHEPRIV	ICACHEPRIV	ETHPRIV	FMACPRIV	CORDICPRIV	CRCPRIV	Res.	Res.	Res.	Res.	Res.	I3C2PRIV	VREFBUFPRIV	LPTIM6PRIV
	Reset value						0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x02C-0x03C	Reserved	Reserved																															
0x040	GTZC1_TZSC_MPCWM1ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x44	GTZC1_TZSC_MPCWM1AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x048	GTZC1_TZSC_MPCWM1BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0

Table 36. GTZC1 TZSC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04C	GTZC1_TZSC_ MPCWM1BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x050	GTZC1_TZSC_ MPCWM2ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x54	GTZC1_TZSC_ MPCWM2AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x058	GTZC1_TZSC_ MPCWM2BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x05C	GTZC1_TZSC_ MPCWM2BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x060	GTZC1_TZSC_ MPCWM3ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x64	GTZC1_TZSC_ MPCWM3AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x068	GTZC1_TZSC_ MPCWM3BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x06C	GTZC1_TZSC_ MPCWM3BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x070	GTZC1_TZSC_ MPCWM4ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x74	GTZC1_TZSC_ MPCWM4AR	Res.	Res.	Res.	Res.	SUBA_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBA_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x078	GTZC1_TZSC_ MPCWM4BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	SEC	Res.	Res.	Res.	Res.	Res.	Res.	SRLOCK	SREN
	Reset value																							0	0							0	0
0x07C	GTZC1_TZSC_ MPCWM4BR	Res.	Res.	Res.	Res.	SUBB_LENGTH[11:0]												Res.	Res.	Res.	Res.	Res.	SUBB_START[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0

Refer to *Table 28: GTZC1 sub-block address offset*.

5.7 GTZC1 TZIC registers

All registers are accessed only by words (32-bit).

5.7.1 GTZC1 TZIC interrupt enable register 1 (GTZC1_TZIC_IER1)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM2IE	DTSIE	UART12IE	UART9IE	UART8IE	UART7IE	DAC1IE	HDMICECIE	USART11IE	USART10IE	USART6IE	CRSIE	I3C1IE	I2C2IE	I2C1IE	UART5IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4IE	USART3IE	USART2IE	SPI3IE	SPI2IE	IWDGIE	WWDGIE	TIM14IE	TIM13IE	TIM12IE	TIM7IE	TIM6IE	TIM5IE	TIM4IE	TIM3IE	TIM2IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LPTIM2IE**: illegal access interrupt enable for LPTIM2
0: interrupt disabled
1: interrupt enabled

Bit 30 **DTSIE**: illegal access interrupt enable for DTS
0: interrupt disabled
1: interrupt enabled

Bit 29 **UART12IE**: illegal access interrupt enable for UART12
0: interrupt disabled
1: interrupt enabled

Bit 28 **UART9IE**: illegal access interrupt enable for UART9
0: interrupt disabled
1: interrupt enabled

Bit 27 **UART8IE**: illegal access interrupt enable for UART8
0: interrupt disabled
1: interrupt enabled

Bit 26 **UART7IE**: illegal access interrupt enable for UART7
0: interrupt disabled
1: interrupt enabled

Bit 25 **DAC1IE**: illegal access interrupt enable for DAC1
0: interrupt disabled
1: interrupt enabled

Bit 24 **HDMICECIE**: illegal access interrupt enable for HDMICEC
0: interrupt disabled
1: interrupt enabled

Bit 23 **USART11IE**: illegal access interrupt enable for USART11
0: interrupt disabled
1: interrupt enabled

Bit 22 **USART10IE**: illegal access interrupt enable for USART10
0: interrupt disabled
1: interrupt enabled

Bit 21 **USART6IE**: illegal access interrupt enable for USART6
0: interrupt disabled
1: interrupt enabled

Bit 20 **CRSIE**: illegal access interrupt enable for CRS
0: interrupt disabled
1: interrupt enabled

Bit 19 **I3C1IE**: illegal access interrupt enable for I3C1
0: interrupt disabled
1: interrupt enabled

Bit 18 **I2C2IE**: illegal access interrupt enable for I2C2
0: interrupt disabled
1: interrupt enabled

Bit 17 **I2C1IE**: illegal access interrupt enable for I2C1
0: interrupt disabled
1: interrupt enabled

Bit 16 **UART5IE**: illegal access interrupt enable for UART5
0: interrupt disabled
1: interrupt enabled

Bit 15 **UART4IE**: illegal access interrupt enable for UART4
0: interrupt disabled
1: interrupt enabled

Bit 14 **USART3IE**: illegal access interrupt enable for USART3
0: interrupt disabled
1: interrupt enabled

Bit 13 **USART2IE**: illegal access interrupt enable for USART2
0: interrupt disabled
1: interrupt enabled

Bit 12 **SPI3IE**: illegal access interrupt enable for SPI3
0: interrupt disabled
1: interrupt enabled

Bit 11 **SPI2IE**: illegal access interrupt enable for SPI2
0: interrupt disabled
1: interrupt enabled

Bit 10 **IWDGIE**: illegal access interrupt enable for IWDG
0: interrupt disabled
1: interrupt enabled

Bit 9 **WWDGIE**: illegal access interrupt enable for WWDG
0: interrupt disabled
1: interrupt enabled

Bit 8 **TIM14IE**: illegal access interrupt enable for TIM14
0: interrupt disabled
1: interrupt enabled

Bit 7 **TIM13IE**: illegal access interrupt enable for TIM13
0: interrupt disabled
1: interrupt enabled

Bit 6 **TIM12IE**: illegal access interrupt enable for TIM12
0: interrupt disabled
1: interrupt enabled

Bit 5 **TIM7IE**: illegal access interrupt enable for TIM7
0: interrupt disabled
1: interrupt enabled

Bit 4 **TIM6IE**: illegal access interrupt enable for TIM6
0: interrupt disabled
1: interrupt enabled

Bit 3 **TIM5IE**: illegal access interrupt enable for TIM5
0: interrupt disabled
1: interrupt enabled

Bit 2 **TIM4IE**: illegal access interrupt enable for TIM4
0: interrupt disabled
1: interrupt enabled

Bit 1 **TIM3IE**: illegal access interrupt enable for TIM3
0: interrupt disabled
1: interrupt enabled

Bit 0 **TIM2IE**: illegal access interrupt enable for TIM2
0: interrupt disabled
1: interrupt enabled

5.7.2 GTZC1 TZIC interrupt enable register 2 (GTZC1_TZIC_IER2)

Address offset: 0x004

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM5IE	LPTIM4IE	LPTIM3IE	LPTIM1IE	I2C4IE	I2C3IE	LPUART1IE	SPI5IE	Res.	Res.	Res.	Res.	USBIE	SAI2IE	SAI1IE	SPI6IE
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI4IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE	SPI1IE	TIM1IE	Res.	Res.	Res.	Res.	Res.	UCPDIE	FDCAN2IE	FDCAN1IE
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bit 31 **LPTIM5IE**: illegal access interrupt enable for LPTIM5
0: interrupt disabled
1: interrupt enabled

Bit 30 **LPTIM4IE**: illegal access interrupt enable for LPTIM4
0: interrupt disabled
1: interrupt enabled

Bit 29 **LPTIM3IE**: illegal access interrupt enable for LPTIM3
0: interrupt disabled
1: interrupt enabled

Bit 28 **LPTIM1IE**: illegal access interrupt enable for LPTIM1
0: interrupt disabled
1: interrupt enabled

Bit 27 **I2C4IE**: illegal access interrupt enable for I2C4
0: interrupt disabled
1: interrupt enabled

Bit 26 **I2C3IE**: illegal access interrupt enable for I2C3
0: interrupt disabled
1: interrupt enabled

Bit 25 **LPUART1IE**: illegal access interrupt enable for LPUART
0: interrupt disabled
1: interrupt enabled

Bit 24 **SPI5IE**: illegal access interrupt enable for SPI5
0: interrupt disabled
1: interrupt enabled

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **USBIE**: illegal access interrupt enable for USB
0: interrupt disabled
1: interrupt enabled

Bit 18 **SAI2IE**: illegal access interrupt enable for SAI2
0: interrupt disabled
1: interrupt enabled

Bit 17 **SAI1IE**: illegal access interrupt enable for SAI1
0: interrupt disabled
1: interrupt enabled

Bit 16 **SPI6IE**: illegal access interrupt enable for SPI6
0: interrupt disabled
1: interrupt enabled

Bit 15 **SPI4IE**: illegal access interrupt enable for SPI4
0: interrupt disabled
1: interrupt enabled

Bit 14 **TIM17IE**: illegal access interrupt enable for TIM17
0: interrupt disabled
1: interrupt enabled

Bit 13 **TIM16IE**: illegal access interrupt enable for TIM16
0: interrupt disabled
1: interrupt enabled

Bit 12 **TIM15IE**: illegal access interrupt enable for TIM15
0: interrupt disabled
1: interrupt enabled

Bit 11 **USART1IE**: illegal access interrupt enable for USART1
0: interrupt disabled
1: interrupt enabled

Bit 10 **TIM8IE**: illegal access interrupt enable for TIM8
0: interrupt disabled
1: interrupt enabled

Bit 9 **SPI1IE**: illegal access interrupt enable for SPI1
0: interrupt disabled
1: interrupt enabled

Bit 8 **TIM1IE**: illegal access interrupt enable for TIM1
0: interrupt disabled
1: interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **UCPDIE**: illegal access interrupt enable for UCPD
0: interrupt disabled
1: interrupt enabled

Bit 1 **FDCAN2IE**: illegal access interrupt enable for FDCAN2
0: interrupt disabled
1: interrupt enabled

Bit 0 **FDCAN1IE**: illegal access interrupt enable for FDCAN1
0: interrupt disabled
1: interrupt enabled

5.7.3 GTZC1 TZIC interrupt enable register 3 (GTZC1_TZIC_IER3)

Address offset: 0x008

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RAMCFGIE	Res.	OCTOSPI1IE	FMCIE	SDMMC2IE	SDMMC1IE	PKAIE	SAESIE	RNGIE	HASHIE	AESIE
					rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCMIIE	ADC12IE	DCACHEIE	ICACHEIE	ETHIE	FMACIE	CORDICIE	CRCIE	Res.	Res.	Res.	Res.	Res.	I3C2IE	VREFBUFIE	LPTIM6IE
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RAMCFGIE**: illegal access interrupt enable for RAMSCFG
0: interrupt disabled
1: interrupt enabled

Bit 25 Reserved, must be kept at reset value.

Bit 24 **OCTOSPI1IE**: illegal access interrupt enable for OCTOSPI1
0: interrupt disabled
1: interrupt enabled

Bit 23 **FMCIE**: illegal access interrupt enable for FMC
0: interrupt disabled
1: interrupt enabled

Bit 22 **SDMMC2IE**: illegal access interrupt enable for SDMMC2
0: interrupt disabled
1: interrupt enabled

Bit 21 **SDMMC1IE**: illegal access interrupt enable for SDMMC1
0: interrupt disabled
1: interrupt enabled

Bit 20 **PKAIE**: illegal access interrupt enable for PKA
0: interrupt disabled
1: interrupt enabled

Bit 19 **SAESIE**: illegal access interrupt enable for SAES
0: interrupt disabled
1: interrupt enabled

Bit 18 **RNGIE**: illegal access interrupt enable for RNG
0: interrupt disabled
1: interrupt enabled

Bit 17 **HASHIE**: illegal access interrupt enable for HASH
0: interrupt disabled
1: interrupt enabled

Bit 16 **AESIE**: illegal access interrupt enable for AES
0: interrupt disabled
1: interrupt enabled

Bit 15 **DCMIIE**: illegal access interrupt enable for DCMI
0: interrupt disabled
1: interrupt enabled

Bit 14 **ADC12IE**: illegal access interrupt enable for ADC1 and ADC2
0: interrupt disabled
1: interrupt enabled

Bit 13 **DCACHEIE**: illegal access interrupt enable for DCACHE
0: interrupt disabled
1: interrupt enabled

Bit 12 **ICACHEIE**: illegal access interrupt enable for ICACHE
0: interrupt disabled
1: interrupt enabled

Bit 11 **ETHIE**: illegal access interrupt enable for register of ETH
0: interrupt disabled
1: interrupt enabled

Bit 10 **FMACIE**: illegal access interrupt enable for FMAC
0: interrupt disabled
1: interrupt enabled

Bit 9 **CORDICIE**: illegal access interrupt enable for CORDIC
0: interrupt disabled
1: interrupt enabled

Bit 8 **CRCIE**: illegal access interrupt enable for CRC
0: interrupt disabled
1: interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **I3C2IE**: illegal access interrupt enable for I3C2
0: interrupt disabled
1: interrupt enabled

Bit 1 **VREFBUFIE**: illegal access interrupt enable for VREFBUF
0: interrupt disabled
1: interrupt enabled

Bit 0 **LPTIM6IE**: illegal access interrupt enable for LPTIM6
0: interrupt disabled
1: interrupt enabled

5.7.4 GTZC1 TZIC interrupt enable register 4 (GTZC1_TZIC_IER4)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure privileged access only.

This register is used to enable interrupt of illegal access.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	MPCBB3_REGIE	SRAM3IE	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	BKPSRAMIE	FMC_MEMIE	OCTOSPI1_MEMIE	TZIC1IE	TZSC1IE
		rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	EXTIIE	RCCIE	PWRIE	TAMPIE	RTCIE	SBSIE	Res.	OTFDEC1IE	FLASHIE	FLASH_REGIE	GPDMA2IE	GPDMA1IE
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **MPCBB3_REGIE**: illegal access interrupt enable for MPCBB3 registers
0: interrupt disabled
1: interrupt enabled

Bit 28 **SRAM3IE**: illegal access interrupt enable for SRAM3
0: interrupt disabled
1: interrupt enabled

Bit 27 **MPCBB2_REGIE**: illegal access interrupt enable for MPCBB2 registers
0: interrupt disabled
1: interrupt enabled

Bit 26 **SRAM2IE**: illegal access interrupt enable for SRAM2
0: interrupt disabled
1: interrupt enabled

Bit 25 **MPCBB1_REGIE**: illegal access interrupt enable for MPCBB1 registers
0: interrupt disabled
1: interrupt enabled

Bit 24 **SRAM1IE**: illegal access interrupt enable for SRAM1
0: interrupt disabled
1: interrupt enabled

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **BKPSRAMIE**: illegal access interrupt enable for MPCWM4 (BKPSRAM) memory bank
0: interrupt disabled
1: interrupt enabled

Bit 19 **FMC_MEMIE**: illegal access interrupt enable for MPCWM2 (FMC_NOR bank), MPCWM3 (FMC_NAND bank and FMC_SDRAM bank 1), and MPCWM4 (FMC_SDRAM bank 2)
0: interrupt disabled
1: interrupt enabled

Bit 18 **OCTOSPI1_MEMIE**: illegal access interrupt enable for MPCWM1 (OCTOSPI1) memory bank
0: interrupt disabled
1: interrupt enabled

Bit 17 **TZIC1IE**: illegal access interrupt enable for GTZC1 TZIC registers
0: interrupt disabled
1: interrupt enabled

Bit 16 **TZSC1IE**: illegal access interrupt enable for GTZC1 TZSC registers
0: interrupt disabled
1: interrupt enabled

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

Bit 11 **EXTIIE**: illegal access interrupt enable for EXTI
0: interrupt disabled
1: interrupt enabled

Bit 10 **RCCIE**: illegal access interrupt enable for RCC
0: interrupt disabled
1: interrupt enabled

Bit 9 **PWRIE**: illegal access interrupt enable for PWR
0: interrupt disabled
1: interrupt enabled

Bit 8 **TAMPIE**: illegal access interrupt enable for TAMP
0: interrupt disabled
1: interrupt enabled

Bit 7 **RTCIE**: illegal access interrupt enable for RTC
0: interrupt disabled
1: interrupt enabled

Bit 6 **SBSIE**: illegal access interrupt enable for SBS
0: interrupt disabled
1: interrupt enabled

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC1IE**: illegal access interrupt enable for OTFDEC1
0: interrupt disabled
1: interrupt enabled

Bit 3 **FLASHIE**: illegal access interrupt enable for FLASH memory
0: interrupt disabled
1: interrupt enabled

Bit 2 **FLASH_REGIE**: illegal access interrupt enable for FLASH registers
0: interrupt disabled
1: interrupt enabled

Bit 1 **GPDMA2IE**: illegal access interrupt enable for GPDMA2
0: interrupt disabled
1: interrupt enabled

Bit 0 **GPDMA1IE**: illegal access interrupt enable for GPDMA1
0: interrupt disabled
1: interrupt enabled

5.7.5 GTZC1 TZIC status register 1 (GTZC1_TZIC_SR1)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM2F	DTSF	UART12F	UART9F	UART8F	UART7F	DAC1F	HDMICECF	USART11F	USART10F	USART6F	CRSF	I3C1F	I2C2F	I2C1F	UART5F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UART4F	USART3F	USART2F	SPI3F	SPI2F	IWDGF	WWDGF	TIM14F	TIM13F	TIM12F	TIM7F	TIM6F	TIM5F	TIM4F	TIM3F	TIM2F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **LPTIM2F**: illegal access flag for LPTIM2
0: no illegal access event
1: illegal access event

Bit 30 **DTSF**: illegal access flag for DTS
0: no illegal access event
1: illegal access event

Bit 29 **UART12F**: illegal access flag for UART12
0: no illegal access event
1: illegal access event

Bit 28 **UART9F**: illegal access flag for UART9
0: no illegal access event
1: illegal access event

Bit 27 **UART8F**: illegal access flag for UART8
0: no illegal access event
1: illegal access event

Bit 26 **UART7F**: illegal access flag for UART7
0: no illegal access event
1: illegal access event

Bit 25 **DAC1F**: illegal access flag for DAC1
0: no illegal access event
1: illegal access event

Bit 24 **HDMICECF**: illegal access flag for HDMICEC
0: no illegal access event
1: illegal access event

Bit 23 **USART11F**: illegal access flag for USART11
0: no illegal access event
1: illegal access event

Bit 22 **USART10F**: illegal access flag for USART10
0: no illegal access event
1: illegal access event

Bit 21 **USART6F**: illegal access flag for USART6
0: no illegal access event
1: illegal access event

Bit 20 **CRSF**: illegal access flag for CRS
0: no illegal access event
1: illegal access event

Bit 19 **I3C1F**: illegal access flag for I3C1
0: no illegal access event
1: illegal access event

Bit 18 **I2C2F**: illegal access flag for I2C2
0: no illegal access event
1: illegal access event

Bit 17 **I2C1F**: illegal access flag for I2C1
0: no illegal access event
1: illegal access event

Bit 16 **UART5F**: illegal access flag for UART5
0: no illegal access event
1: illegal access event

Bit 15 **UART4F**: illegal access flag for UART4
0: no illegal access event
1: illegal access event

Bit 14 **USART3F**: illegal access flag for USART3
0: no illegal access event
1: illegal access event

Bit 13 **USART2F**: illegal access flag for USART2
0: no illegal access event
1: illegal access event

Bit 12 **SPI3F**: illegal access flag for SPI3
0: no illegal access event
1: illegal access event

Bit 11 **SPI2F**: illegal access flag for SPI2
0: no illegal access event
1: illegal access event

Bit 10 **IWDGF**: illegal access flag for IWDG
0: no illegal access event
1: illegal access event

Bit 9 **WWDGF**: illegal access flag for WWDG
0: no illegal access event
1: illegal access event

Bit 8 **TIM14F**: illegal access flag for TIM14
0: no illegal access event
1: illegal access event

Bit 7 **TIM13F**: illegal access flag for TIM13
0: no illegal access event
1: illegal access event

Bit 6 **TIM12F**: illegal access flag for TIM12
0: no illegal access event
1: illegal access event

Bit 5 **TIM7F**: illegal access flag for TIM7
0: no illegal access event
1: illegal access event

Bit 4 **TIM6F**: illegal access flag for TIM6
0: no illegal access event
1: illegal access event

Bit 3 **TIM5F**: illegal access flag for TIM5
0: no illegal access event
1: illegal access event

Bit 2 **TIM4F**: illegal access flag for TIM4
0: no illegal access event
1: illegal access event

Bit 1 **TIM3F**: illegal access flag for TIM3
0: no illegal access event
1: illegal access event

Bit 0 **TIM2F**: illegal access flag for TIM2
0: no illegal access event
1: illegal access event

5.7.6 GTZC1 TZIC status register 2 (GTZC1_TZIC_SR2)

Address offset: 0x014

Reset value: 0x0000 0000

Secure privileged access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPTIM5F	LPTIM4F	LPTIM3F	LPTIM1F	I2C4F	I2C3F	LPUART1F	SPI5F	Res.	Res.	Res.	Res.	USBF	SAI2F	SAI1F	SPI6F
r	r	r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI4F	TIM17F	TIM16F	TIM15F	USART1F	TIM8F	SPI1F	TIM1F	Res.	Res.	Res.	Res.	Res.	UCPDF	FDCAN2F	FDCAN1F
r	r	r	r	r	r	r	r						r	r	r

Bit 31 **LPTIM5F**: illegal access flag for LPTIM5
0: no illegal access event
1: illegal access event

Bit 30 **LPTIM4F**: illegal access flag for LPTIM4
0: no illegal access event
1: illegal access event

Bit 29 **LPTIM3F**: illegal access flag for LPTIM3
0: no illegal access event
1: illegal access event

Bit 28 **LPTIM1F**: illegal access flag for LPTIM1
0: no illegal access event
1: illegal access event

Bit 27 **I2C4F**: illegal access flag for I2C4
0: no illegal access event
1: illegal access event

Bit 26 **I2C3F**: illegal access flag for I2C3
0: no illegal access event
1: illegal access event

Bit 25 **LPUART1F**: illegal access flag for LPUART
0: no illegal access event
1: illegal access event

Bit 24 **SPI5F**: illegal access flag for SPI5
0: no illegal access event
1: illegal access event

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **USBF**: illegal access flag for USB
0: no illegal access event
1: illegal access event

Bit 18 **SAI2F**: illegal access flag for SAI2
0: no illegal access event
1: illegal access event

Bit 17 **SAI1F**: illegal access flag for SAI1
0: no illegal access event
1: illegal access event

Bit 16 **SPI6F**: illegal access flag for SPI6
0: no illegal access event
1: illegal access event

Bit 15 **SPI4F**: illegal access flag for SPI4
0: no illegal access event
1: illegal access event

Bit 14 **TIM17F**: illegal access flag for TIM17
0: no illegal access event
1: illegal access event

Bit 13 **TIM16F**: illegal access flag for TIM16
0: no illegal access event
1: illegal access event

Bit 12 **TIM15F**: illegal access flag for TIM15
0: no illegal access event
1: illegal access event

Bit 11 **USART1F**: illegal access flag for USART1
0: no illegal access event
1: illegal access event

Bit 10 **TIM8F**: illegal access flag for TIM8
0: no illegal access event
1: illegal access event

Bit 9 **SPI1F**: illegal access flag for SPI1
0: no illegal access event
1: illegal access event

Bit 8 **TIM1F**: illegal access flag for TIM1
0: no illegal access event
1: illegal access event

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **UCPDF**: illegal access flag for UCPD
0: no illegal access event
1: illegal access event

Bit 1 **FDCAN2F**: illegal access flag for FDCAN2
0: no illegal access event
1: illegal access event

Bit 0 **FDCAN1F**: illegal access flag for FDCAN1
0: no illegal access event
1: illegal access event

5.7.7 GTZC1 TZIC status register 3 (GTZC1_TZIC_SR3)

Address offset: 0x018

Reset value: 0x0000 0000

Secure privileged access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RAMCFGF	Res.	OCTOSPI1F	FMCF	SDMMC2F	SDMMC1F	PKAF	SAESF	RNGF	HASHF	AESF
					r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCMIF	ADC12F	DCACHEF	ICACHEF	ETHF	FMACF	CORDICF	CRCF	Res.	Res.	Res.	Res.	Res.	I3C2F	VREFBUFF	LPTIM6F
r	r	r	r	r	r	r	r						r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **RAMCFGF**: illegal access flag for RAMSCFG
0: no illegal access event
1: illegal access event

Bit 25 Reserved, must be kept at reset value.

Bit 24 **OCTOSPI1F**: illegal access flag for OCTOSPI1
0: no illegal access event
1: illegal access event

Bit 23 **FMCF**: illegal access flag for FMC
0: no illegal access event
1: illegal access event

Bit 22 **SDMMC2F**: illegal access flag for SDMMC2
0: no illegal access event
1: illegal access event

Bit 21 **SDMMC1F**: illegal access flag for SDMMC1
0: no illegal access event
1: illegal access event

Bit 20 **PKAF**: illegal access flag for PKA
0: no illegal access event
1: illegal access event

Bit 19 **SAESF**: illegal access flag for SAES
0: no illegal access event
1: illegal access event

Bit 18 **RNGF**: illegal access flag for RNG
0: no illegal access event
1: illegal access event

Bit 17 **HASHF**: illegal access flag for HASH
0: no illegal access event
1: illegal access event

Bit 16 **AESF**: illegal access flag for AES
0: no illegal access event
1: illegal access event

Bit 15 **DCMIF**: illegal access flag for DCMI
0: no illegal access event
1: illegal access event

Bit 14 **ADC12F**: illegal access flag for ADC1 and ADC2
0: no illegal access event
1: illegal access event

Bit 13 **DCACHEF**: illegal access flag for DCACHE
0: no illegal access event
1: illegal access event

Bit 12 **ICACHEF**: illegal access flag for ICACHE
0: no illegal access event
1: illegal access event

Bit 11 **ETHF**: illegal access flag for register of ETH
0: no illegal access event
1: illegal access event

Bit 10 **FMACF**: illegal access flag for FMAC
0: no illegal access event
1: illegal access event

Bit 9 **CORDICF**: illegal access flag for CORDIC
0: no illegal access event
1: illegal access event

Bit 8 **CRCF**: illegal access flag for CRC
0: no illegal access event
1: illegal access event

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **I3C2F**: illegal access flag for I3C2
0: no illegal access event
1: illegal access event

Bit 1 **VREFBUFF**: illegal access flag for VREFBUF
0: no illegal access event
1: illegal access event

Bit 0 **LPTIM6F**: illegal access flag for LPTIM6
0: no illegal access event
1: illegal access event

5.7.8 GTZC1 TZIC status register 4 (GTZC1_TZIC_SR4)

Address offset: 0x01C

Reset value: 0x0000 0000

Secure privileged access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	MPCBB3_REGF	SRAM3F	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	BKPSRAMF	FMC_MEMF	OCTOSPI1_MEMF	TZIC1F	TZSC1F
		r	r	r	r	r	r				r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	EXTIF	RCCF	PWRF	TAMPF	RTCF	SBSF	Res.	OTFDEC1F	FLASHF	FLASH_REGF	GPDMA2F	GPDMA1F
				r	r	r	r	r	r		r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **MPCBB3_REGF**: illegal access flag for MPCBB3 registers
0: no illegal access event
1: illegal access event

Bit 28 **SRAM3F**: illegal access flag for SRAM3
0: no illegal access event
1: illegal access event

Bit 27 **MPCBB2_REGF**: illegal access flag for MPCBB2 registers
0: no illegal access event
1: illegal access event

Bit 26 **SRAM2F**: illegal access flag for SRAM2
0: no illegal access event
1: illegal access event

Bit 25 **MPCBB1_REGF**: illegal access flag for MPCBB1 registers
0: no illegal access event
1: illegal access event

Bit 24 **SRAM1F**: illegal access flag for SRAM1
0: no illegal access event
1: illegal access event

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **BKPSRAMF**: illegal access flag for MPCWM4 (BKPSRAM) memory bank
0: no illegal access event
1: illegal access event

Bit 19 **FMC_MEMF**: illegal access flag for MPCWM2 (FMC_NOR bank), MPCWM3 (FMC_NAND bank and FMC_SDRAM bank 1), and MPCWM4 (FMC_SDRAM bank 2)
0: no illegal access event
1: illegal access event

Bit 18 **OCTOSPI1_MEMF**: illegal access flag for MPCWM1 (OCTOSPI1) memory bank
0: no illegal access event
1: illegal access event

Bit 17 **TZIC1F**: illegal access flag for GTZC1 TZIC registers
0: no illegal access event
1: illegal access event

Bit 16 **TZSC1F**: illegal access flag for GTZC1 TZSC registers
0: no illegal access event
1: illegal access event

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

Bit 11 **EXTIF**: illegal access flag for EXTI
0: no illegal access event
1: illegal access event

Bit 10 **RCCF**: illegal access flag for RCC
0: no illegal access event
1: illegal access event

Bit 9 **PWRF**: illegal access flag for PWR
0: no illegal access event
1: illegal access event

Bit 8 **TAMPF**: illegal access flag for TAMP
0: no illegal access event
1: illegal access event

Bit 7 **RTCF**: illegal access flag for RTC
0: no illegal access event
1: illegal access event

Bit 6 **SBSF**: illegal access flag for SBS
0: no illegal access event
1: illegal access event

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTFDEC1F**: illegal access flag for OTFDEC1
0: no illegal access event
1: illegal access event

Bit 3 **FLASHF**: illegal access flag for FLASH memory
0: no illegal access event
1: illegal access event

Bit 2 **FLASH_REGF**: illegal access flag for FLASH registers
0: no illegal access event
1: illegal access event

Bit 1 **GPDMA2F**: illegal access flag for GPDMA2
0: no illegal access event
1: illegal access event

Bit 0 **GPDMA1F**: illegal access flag for GPDMA1
0: no illegal access event
1: illegal access event

5.7.9 GTZC1 TZIC flag clear register 1 (GTZC1_TZIC_FCR1)

Address offset: 0x020

Reset value: 0x0000 0000

Secure privileged access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLPTIM2F	CDTSF	CUART12F	CUART9F	CUART8F	CUART7F	CDAC1F	CHDMICECF	CUSART11F	CUSART10F	CUSART6F	CCRSF	CI3C1F	CI2C2F	CI2C1F	CUART5F
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CUART4F	CUSART3F	CUSART2F	CSPI3F	CSPI2F	CIWDGF	CWWDGF	CTIM14F	CTIM13F	CTIM12F	CTIM7F	CTIM6F	CTIM5F	CTIM4F	CTIM3F	CTIM2F
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **CLPTIM2F**: clear the illegal access flag for LPTIM2
0: no action
1: status flag cleared

Bit 30 **CDTSF**: clear the illegal access flag for DTS
0: no action
1: status flag cleared

Bit 29 **CUART12F**: clear the illegal access flag for UART12
0: no action
1: status flag cleared

Bit 28 **CUART9F**: clear the illegal access flag for UART9
0: no action
1: status flag cleared

Bit 27 **CUART8F**: clear the illegal access flag for UART8
0: no action
1: status flag cleared

Bit 26 **CUART7F**: clear the illegal access flag for UART7
0: no action
1: status flag cleared

Bit 25 **CDAC1F**: clear the illegal access flag for DAC1
0: no action
1: status flag cleared

Bit 24 **CHDMICECF**: clear the illegal access flag for HDMICEC
0: no action
1: status flag cleared

Bit 23 **CUSART11F**: clear the illegal access flag for USART11
0: no action
1: status flag cleared

Bit 22 **CUSART10F**: clear the illegal access flag for USART10
0: no action
1: status flag cleared

Bit 21 **CUSART6F**: clear the illegal access flag for USART6
0: no action
1: status flag cleared

Bit 20 **CCRSF**: clear the illegal access flag for CRS
0: no action
1: status flag cleared

Bit 19 **CI3C1F**: clear the illegal access flag for I3C1
0: no action
1: status flag cleared

Bit 18 **CI2C2F**: clear the illegal access flag for I2C2
0: no action
1: status flag cleared

Bit 17 **CI2C1F**: clear the illegal access flag for I2C1
0: no action
1: status flag cleared

Bit 16 **CUART5F**: clear the illegal access flag for UART5
0: no action
1: status flag cleared

Bit 15 **CUART4F**: clear the illegal access flag for UART4
0: no action
1: status flag cleared

Bit 14 **CUSART3F**: clear the illegal access flag for USART3
0: no action
1: status flag cleared

Bit 13 **CUSART2F**: clear the illegal access flag for USART2
0: no action
1: status flag cleared

Bit 12 **CSPI3F**: clear the illegal access flag for SPI3
0: no action
1: status flag cleared

Bit 11 **CSPI2F**: clear the illegal access flag for SPI2
0: no action
1: status flag cleared

Bit 10 **CIWDGF**: clear the illegal access flag for IWDG
0: no action
1: status flag cleared

Bit 9 **CWWDGF**: clear the illegal access flag for WWDG
0: no action
1: status flag cleared

Bit 8 **CTIM14F**: clear the illegal access flag for TIM14
0: no action
1: status flag cleared

Bit 7 **CTIM13F**: clear the illegal access flag for TIM13
0: no action
1: status flag cleared

Bit 6 **CTIM12F**: clear the illegal access flag for TIM12
0: no action
1: status flag cleared

Bit 5 **CTIM7F**: clear the illegal access flag for TIM7
0: no action
1: status flag cleared

Bit 4 **CTIM6F**: clear the illegal access flag for TIM6
0: no action
1: status flag cleared

Bit 3 **CTIM5F**: clear the illegal access flag for TIM5
0: no action
1: status flag cleared

Bit 2 **CTIM4F**: clear the illegal access flag for TIM4
0: no action
1: status flag cleared

Bit 1 **CTIM3F**: clear the illegal access flag for TIM3
0: no action
1: status flag cleared

Bit 0 **CTIM2F**: clear the illegal access flag for TIM2
0: no action
1: status flag cleared

5.7.10 GTZC1 TZIC flag clear register 2 (GTZC1_TZIC_FCR2)

Address offset: 0x024

Reset value: 0x0000 0000

Secure privileged access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLPTIM5F	CLPTIM4F	CLPTIM3F	CLPTIM1F	CI2C4F	CI2C3F	CLPUART1F	CSPI5F	Res.	Res.	Res.	Res.	CUSBF	CSAI2F	CSAI1F	CSPI6F
w	w	w	w	w	w	w	w					w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSPI4F	CTIM17F	CTIM16F	CTIM15F	CUSART1F	CTIM8F	CSPI1F	CTIM1F	Res.	Res.	Res.	Res.	Res.	CUCPDF	CFDCAN2F	CFDCAN1F
w	w	w	w	w	w	w	w						w	w	w

Bit 31 **CLPTIM5F**: clear the illegal access flag for LPTIM5
0: no action
1: status flag cleared

Bit 30 **CLPTIM4F**: clear the illegal access flag for LPTIM4
0: no action
1: status flag cleared

Bit 29 **CLPTIM3F**: clear the illegal access flag for LPTIM3
0: no action
1: status flag cleared

Bit 28 **CLPTIM1F**: clear the illegal access flag for LPTIM1
0: no action
1: status flag cleared

Bit 27 **CI2C4F**: clear the illegal access flag for I2C4
0: no action
1: status flag cleared

Bit 26 **CI2C3F**: clear the illegal access flag for I2C3
0: no action
1: status flag cleared

Bit 25 **CLPUART1F**: clear the illegal access flag for LPUART
0: no action
1: status flag cleared

Bit 24 **CSPI5F**: clear the illegal access flag for SPI5
0: no action
1: status flag cleared

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CUSBF**: clear the illegal access flag for USB
0: no action
1: status flag cleared

Bit 18 **CSAI2F**: clear the illegal access flag for SAI2
0: no action
1: status flag cleared

Bit 17 **CSAI1F**: clear the illegal access flag for SAI1
0: no action
1: status flag cleared

Bit 16 **CSPI6F**: clear the illegal access flag for SPI6
0: no action
1: status flag cleared

Bit 15 **CSPI4F**: clear the illegal access flag for SPI4
0: no action
1: status flag cleared

Bit 14 **CTIM17F**: clear the illegal access flag for TIM17
0: no action
1: status flag cleared

Bit 13 **CTIM16F**: clear the illegal access flag for TIM16
0: no action
1: status flag cleared

Bit 12 **CTIM15F**: clear the illegal access flag for TIM15
0: no action
1: status flag cleared

Bit 11 **CUSART1F**: clear the illegal access flag for USART1
0: no action
1: status flag cleared

Bit 10 **CTIM8F**: clear the illegal access flag for TIM8
0: no action
1: status flag cleared

Bit 9 **CSPI1F**: clear the illegal access flag for SPI1
0: no action
1: status flag cleared

Bit 8 **CTIM1F**: clear the illegal access flag for TIM1
0: no action
1: status flag cleared

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CUCPDF**: clear the illegal access flag for UCPD
0: no action
1: status flag cleared

Bit 1 **CFDCAN2F**: clear the illegal access flag for FDCAN2
0: no action
1: status flag cleared

Bit 0 **CFDCAN1F**: clear the illegal access flag for FDCAN1
0: no action
1: status flag cleared

5.7.11 GTZC1 TZIC flag clear register 3 (GTZC1_TZIC_FCR3)

Address offset: 0x028

Reset value: 0x0000 0000

Secure privilege access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CRAMCFGF	Res.	COCTOSPI1F	CFMCF	CSDMMC2F	CSDMMC1F	CPKAF	CSAESF	CRNGF	CHASHF	CAESF
					w		w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDCMIF	CADC12F	CDCACHEF	CICACHEF	CETHF	CFMACF	CCORDICF	CCRCF	Res.	Res.	Res.	Res.	Res.	CI3C2F	CVREFBUFF	CLPTIM6F
w	w	w	w	w	w	w	w						w	w	w

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **CRAMCFGF**: clear illegal access flag for RAMSCFG
0: no action
1: status flag cleared

Bit 25 Reserved, must be kept at reset value.

Bit 24 **COCTOSPI1F**: clear illegal access flag for OCTOSPI1
0: no action
1: status flag cleared

Bit 23 **CFMCF**: clear illegal access flag for FMC
0: no action
1: status flag cleared

Bit 22 **CSDMMC2F**: clear illegal access flag for SDMMC2
0: no action
1: status flag cleared

Bit 21 **CSDMMC1F**: clear illegal access flag for SDMMC1
0: no action
1: status flag cleared

Bit 20 **CPKAF**: clear illegal access flag for PKA
0: no action
1: status flag cleared

Bit 19 **CSAESF**: clear illegal access flag for SAES
0: no action
1: status flag cleared

Bit 18 **CRNGF**: clear illegal access flag for RNG
0: no action
1: status flag cleared

Bit 17 **CHASHF**: clear illegal access flag for HASH
0: no action
1: status flag cleared

Bit 16 **CAESF**: clear illegal access flag for AES
0: no action
1: status flag cleared

Bit 15 **CDCMIF**: clear illegal access flag for DCMI
0: no action
1: status flag cleared

Bit 14 **CADC12F**: clear illegal access flag for ADC1 and ADC2
0: no action
1: status flag cleared

Bit 13 **CDCACHEF**: clear illegal access flag for DCACHE
0: no action
1: status flag cleared

Bit 12 **CICACHEF**: clear illegal access flag for ICACHE
0: no action
1: status flag cleared

Bit 11 **CETHF**: clear illegal access flag for register of ETH
0: no action
1: status flag cleared

Bit 10 **CFMACF**: clear illegal access flag for FMAC
0: no action
1: status flag cleared

Bit 9 **CCORDICF**: clear illegal access flag for CORDIC
0: no action
1: status flag cleared

Bit 8 **CCRCF**: clear illegal access flag for CRC
0: no action
1: status flag cleared

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **CI3C2F**: clear illegal access flag for I3C2
0: no action
1: status flag cleared

Bit 1 **CVREFBUFF**: clear illegal access flag for VREFBUF
0: no action
1: status flag cleared

Bit 0 **CLPTIM6F**: clear illegal access flag for LPTIM6
0: no action
1: status flag cleared

5.7.12 GTZC1 TZIC flag clear register 4 (GTZC1_TZIC_FCR4)

Address offset: 0x02C

Reset value: 0x0000 0000

Secure privilege access only.

Refer to the device datasheet for the peripheral availability. If not present, consider the associated bit as reserved, and keep it at reset value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CMPCBB3_REGF	CSRAM3F	CMPCBB2_REGF	CSRAM2F	CMPCBB1_REGF	CSRAM1F	Res.	Res.	Res.	CBKPSRAMF	CFMC_MEMF	COCTOSPI1_MEMF	CTZIC1F	CTZSC1F
		w	w	w	w	w	w				w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CEXTIF	CRCCF	CPWRF	CTAMPF	CRTCF	CSBSF	Res.	COTFDEC1F	CFLASHF	CFLASH_REGF	CGPDMA2F	CGPDMA1F
				w	w	w	w	w	w		w	w	w	w	w

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CMPCBB3_REGF**: clear the illegal access flag for MPCBB3 registers
0: no action
1: status flag cleared

Bit 28 **CSRAM3F**: clear the illegal access flag for SRAM3
0: no action
1: status flag cleared

Bit 27 **CMPCBB2_REGF**: clear the illegal access flag for MPCBB2 registers
0: no action
1: status flag cleared

Bit 26 **CSRAM2F**: clear the illegal access flag for SRAM2
0: no action
1: status flag cleared

Bit 25 **CMPCBB1_REGF**: clear the illegal access flag for MPCBB1 registers
0: no action
1: status flag cleared

Bit 24 **CSRAM1F**: clear the illegal access flag for SRAM1
0: no action
1: status flag cleared

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CBKPSRAMF**: clear the illegal access flag for MPCWM4 (BKPSRAM) memory bank
0: no action
1: status flag cleared

Bit 19 **CFMC_MEMF**: clear the illegal access flag for MPCWM2 (FMC_NOR bank), MPCWM3 (FMC_NAND bank and FMC_SDRAM bank 1), and MPCWM4 (FMC_SDRAM bank 2)
0: no action
1: status flag cleared

Bit 18 **COCTOSPI1_MEMF**: clear the illegal access flag for MPCWM1 (OCTOSPI1) memory bank
0: no action
1: status flag cleared

Bit 17 **CTZIC1F**: clear the illegal access flag for GTZC1 TZIC registers
0: no action
1: status flag cleared

Bit 16 **CTZSC1F**: clear the illegal access flag for GTZC1 TZSC registers
0: no action
1: status flag cleared

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

Bit 11 **CEXTIF**: clear the illegal access flag for EXTI
0: no action
1: status flag cleared

Bit 10 **CRCCF**: clear the illegal access flag for RCC
0: no action
1: status flag cleared

Bit 9 **CPWRF**: clear the illegal access flag for PWR
0: no action
1: status flag cleared

Bit 8 **CTAMPF**: clear the illegal access flag for TAMP
0: no action
1: status flag cleared

Bit 7 **CRTCF**: clear the illegal access flag for RTC
0: no action
1: status flag cleared

Bit 6 **CSBSF**: clear the illegal access flag for SBS
0: no action
1: status flag cleared

Bit 5 Reserved, must be kept at reset value.

Bit 4 **COTFDEC1F**: clear the illegal access flag for OTFDEC1
0: no action
1: status flag cleared

Bit 3 **CFLASHF**: clear the illegal access flag for FLASH memory
0: no action
1: status flag cleared

Bit 2 **CFLASH_REGF**: clear the illegal access flag for FLASH registers
0: no action
1: status flag cleared

Bit 1 **CGPDMA2F**: clear the illegal access flag for GPDMA2
0: no action
1: status flag cleared

Bit 0 **CGPDMA1F**: clear the illegal access flag for GPDMA1
0: no action
1: status flag cleared

5.7.13 GTZC1 TZIC register map

Table 37. GTZC1 TZIC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_TZIC_IER1	LPTIM2IE	DTSIE	UART12IE	UART9IE	UART8IE	UART7IE	DAC1IE	HDMICECIE	USART11IE	USART10IE	USART6IE	CRSIE	I3C1IE	I2C2IE	I2C1IE	UART5IE	UART4IE	USART3IE	USART2IE	SPI3IE	SPI2IE	IWDGIE	WWDGIE	TIM14IE	TIM13IE	TIM112IE	TIM7IE	TIM6IE	TIM5IE	TIM4IE	TIM3IE	TIM2IE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	GTZC1_TZIC_IER2	LPTIM5IE	LPTIM4IE	LPTIM3IE	LPTIM1IE	I2C4IE	I2C3IE	LPUART1IE	SPI5IE	Res.	Res.	Res.	Res.	USBIE	SAI2IE	SAI1IE	SPI6IE	SPI4IE	TIM17IE	TIM16IE	TIM15IE	USART1IE	TIM8IE	SPI1IE	TIM1IE	Res.	Res.	Res.	Res.	Res.	Res.	UCPDIE	FDCAN2IE	FDCAN1IE
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0							0	0	0
0x008	GTZC1_TZIC_IER3	Res.	Res.	Res.	Res.	Res.	RAMCFGIE	Res.	OCTOSPI1IE	FMCIE	SDMMC2IE	SDMMC1IE	PKAIE	SAESIE	RNGIE	HASHIE	AESIE	DCMIIE	ADC12IE	DCACHEIE	ICACHEIE	ETHIE	FMACIE	CORDICIE	CRCIE	Res.	Res.	Res.	Res.	Res.	Res.	I3C2IE	VREFBUFIE	LPTIM6IE
	Reset value						0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0
0x00C	GTZC1_TZIC_IER4	Res.	Res.	MPCBB3_REGIE	SRAM3IE	MPCBB2_REGIE	SRAM2IE	MPCBB1_REGIE	SRAM1IE	Res.	Res.	Res.	BKPSRAMIE	FMC_MEMIE	OCTOSPI1_MEMIE	TZIC1IE	TZSC1IE	Res.	Res.	Res.	Res.	EXTIIE	RCCIE	PWRIE	TAMPIE	RTCIE	SBSIE	Res.	OTFDEC1IE	FLASHIE	FLASH_REGIE	GPDMA2IE	GPDMA1IE
	Reset value			0	0	0	0	0	0				0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0	0
0x010	GTZC1_TZIC_SR1	LPTIM2F	DTSF	UART12F	UART9F	UART8F	UART7F	DAC1F	HDMICECF	USART11F	USART10F	USART6F	CRSF	I3C1F	I2C2F	I2C1F	UART5F	UART4F	USART3F	USART2F	SPI3F	SPI2F	IWDGF	WWDGF	TIM14F	TIM13F	TIM112F	TIM7F	TIM6F	TIM5F	TIM4F	TIM3F	TIM2F
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	GTZC1_TZIC_SR2	LPTIM5F	LPTIM4F	LPTIM3F	LPTIM1F	I2C4F	I2C3F	LPUART1F	SPI5F	Res.	Res.	Res.	Res.	USBF	SAI2F	SAI1F	SPI6F	SPI4F	TIM17F	TIM16F	TIM15F	USART1F	TIM8F	SPI1F	TIM1F	Res.	Res.	Res.	Res.	Res.	Res.	UCPDF	FDCAN2F	FDCAN1F
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0							0	0	0
0x018	GTZC1_TZIC_SR3	Res.	Res.	Res.	Res.	Res.	RAMCFGF	Res.	OCTOSPI1F	FMCF	SDMMC2F	SDMMC1F	PKAF	SAESF	RNGF	HASHF	AESF	DCMIF	ADC12F	DCACHEF	ICACHEF	ETHF	FMACF	CORDICF	CRCF	Res.	Res.	Res.	Res.	Res.	Res.	I3C2F	VREFBUFF	LPTIM6F
	Reset value						0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0	0
0x01C	GTZC1_TZIC_SR4	Res.	Res.	MPCBB3_REGF	SRAM3F	MPCBB2_REGF	SRAM2F	MPCBB1_REGF	SRAM1F	Res.	Res.	Res.	BKPSRAMF	FMC_MEMF	OCTOSPI1_MEMF	TZIC1F	TZSC1F	Res.	Res.	Res.	Res.	EXTIF	RCCF	PWRF	TAMPF	RTCF	SBSF	Res.	OTFDEC1F	FLASHF	FLASH_REGF	GPDMA2F	GPDMA1F
	Reset value			0	0	0	0	0	0				0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0	0
0x020	GTZC1_TZIC_FCR1	CLPTIM2F	CDTSF	CUART12F	CUART9F	CUART8F	CUART7F	CDAC1F	CHDMICECF	CUSART11F	CUSART10F	CUSART6F	CCRSF	CI3C1F	CI2C2F	CI2C1F	CUART5F	CUART4F	CUSART3F	CUSART2F	CSPI3F	CSPI2F	CIWDGF	CWWDGF	CTIM14F	CTIM13F	CTIM112F	CTIM7F	CTIM6F	CTIM5F	CTIM4F	CTIM3F	CTIM2F
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 37. GTZC1 TZIC register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	GTZC1_TZIC_FCR2	CLPTIM5F	CLPTIM4F	CLPTIM3F	CLPTIM1F	CI2C4F	CI2C3F	CLPUART1F	CSPI5F	Res.	Res.	Res.	Res.	CUSBF	CSAI2F	CSAI1F	CSPI6F	CSPI4F	CTIM17F	CTIM16F	CTIM15F	CUSART1F	CTIM8F	CSPI1F	CTIM1F	Res.	Res.	Res.	Res.	Res.	CUCPDF	CFDCAN2F	CFDCAN1F
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x028	GTZC1_TZIC_FCR3	Res.	Res.	Res.	Res.	Res.	CRAMCFGF	Res.	COCTOSPI1F	CFMCF	CSDMMC2F	CSDMMC1F	CPKAF	CSAESF	CRNGF	CHASHF	CAESF	CDCMIF	CADC12F	CDCACHEF	CICACHEF	CETHF	CFMACF	CCORDICF	CCRCF	Res.	Res.	Res.	Res.	Res.	Res.	CI3C2F	CVREFBUFF	CLPTIM6F
	Reset value						0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x02C	GTZC1_TZIC_FCR4	Res.	Res.	CMPCBB3_REGF	CSRAM3F	CMPCBB2_REGF	CSRAM2F	CMPCBB1_REGF	CSRAM1F	Res.	Res.	Res.	CBKPSRAMF	CFMC_MEMF	COCTOSPI1_MEMF	CTZIC1F	CTZSC1F	Res.	Res.	Res.	Res.	CEXTIF	CRCCF	CPWRF	CTAMPF	CRTCF	CSBSF	Res.	COTFDEC1F	CFLASHF	CFLASH_REGF	CGPDMA2F	CGPDMA1F
	Reset value			0	0	0	0	0	0				0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0

Refer to *Table 28: GTZC1 sub-block address offset*.

5.8 GTZC1 MPCBBz registers (z = 1 to 3)

All registers are accessed only by words (32-bit).

5.8.1 GTZC1 SRAMz MPCBB control register (GTZC1_MPCBBz_CR) (z = 1 to 3)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged access only.

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

Bit 31 **SRWILADIS**: secure read/write illegal access disable

This bit disables the detection of an illegal access when a secure read/write transaction access a non-secure blocks of the block-based SRAM (secure fetch on non-secure block is always considered illegal).

0: enabled, secure read/write access not allowed on non-secure SRAM block

1: disabled, secure read/write access allowed on non-secure SRAM block

Bit 30 **INVSECSTATE**: SRAMx clocks security state

This bit is used to define the internal SRAMs clocks control in RCC as secure or not.

0: SRAMs clocks are secure if a secure area exists in the MPCBB. It is non secure if there is no secure area.

1: SRAMs clocks are non-secure even if a secure area exists in the MPCBB, and secure even if no secure block is set in the MPCBB.

Bits 29:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: lock the control register of the MPCBB until next reset

This bit is cleared by default, and once set, it cannot be reset until system reset.

0: control register not locked

1: control register locked

5.8.2 GTZC1 SRAMz MPCBB configuration lock register 1 (GTZC1_MPCBBz_CFGLOCK1) (z = 1 to 3)

Address offset: 0x010

Reset value: 0x0000 0000

Secure privileged access only.

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

Bits 31:0 **SPLCKy**: Security/privilege configuration lock for super-block (y = 31 to 0)
This bit is set by software and can be cleared only by system reset.
0: GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy can be written.
1: Writes to GTZC1_MPCBBz_SECCFGRy and GTZC1_MPCBBz_PRIVCFGRy are ignored

5.8.3 GTZC1 SRAMz MPCBB security configuration for super-block x register (GTZC1_MPCBBz_SECCFGRx) (z = 1 to 3)

Address offset: 0x100 + 0x4 * x, (x = 0 to 31)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0000 0000 when TZEN = 0xC3.

Write access to this register is secure only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECy**: Security configuration for block y (y = 31 to 0)
0: Non-secure access only to block y, belonging to super-block x. Secure access is also allowed if the SRWILADIS bit is set in GTZC1_MPCBBz_CR.
1: Secure access only to block y, belonging to super-block x.
Unprivileged write to this bit is ignored if PRIVy bit is set in GTZC1_MPCBBz_PRIVCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCK.

5.8.4 GTZC1 SRAMz MPCBB privileged configuration for super-block x register (GTZC1_MPCBBz_PRIVCFGRx) (z = 1 to 3)

Address offset: 0x200 + 0x4 * x, (x = 0 to 31)

Reset value: 0xFFFF FFFF

The given reset value is valid when TZEN = 0xB4. The reset value is 0x0000 0000 when TZEN = 0xC3. Write access to this register is privileged only. Any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVy**: Privileged configuration for block y, belonging to super-block x (y = 31 to 0).
0: Privileged and unprivileged access to block y, belonging to super-block x
1: Only privileged access to block y, belonging to super-block x
Non-secure write to this bit is ignored if SECy bit is set in GTZC1_MPCBBz_SECCFGRx.
Writes are ignored if SPLCKx bit is set in GTZC1_MPCBBz_CFGLOCK.

5.8.5 GTZC1 MPCBBz register map (z = 1 to 3)

Table 38. GTZC1 MPCBBz register map and reset values (z = 1 to 3)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GTZC1_ MPCBBz_CR	SRWILADIS	INVSECSTATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value	0	0																														0
0x004-0x00C	Reserved	Reserved																															
0x010	GTZC1_MPCBBz_ CFGLOCK1	SPLCK31	SPLCK30	SPLCK29	SPLCK28	SPLCK27	SPLCK26	SPLCK25	SPLCK24	SPLCK23	SPLCK22	SPLCK21	SPLCK20	SPLCK19	SPLCK18	SPLCK17	SPLCK16	SPLCK15	SPLCK14	SPLCK13	SPLCK12	SPLCK11	SPLCK10	SPLCK9	SPLCK8	SPLCK7	SPLCK6	SPLCK5	SPLCK4	SPLCK3	SPLCK2	SPLCK1	SPLCK0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014-0x0FC	Reserved	Reserved																															
0x100 + 0x04 *x (x = 0 to 31)	GTZC1_MPCBBz_ SECCFGRx	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x180-0x1FC	Reserved	Reserved																															
0x200 + 0x04 *x (x = 0 to 31)	GTZC1_MPCBBz_ PRIVCFGRx	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Table 28: GTZC1 sub-block address offset*.

6 RAMs configuration controller (RAMCFG)

6.1 Introduction

The RAMCFG configures the features of the internal SRAMs (SRAM1, SRAM2, SRAM3, and BKPSRAM).

6.2 RAMCFG main features

The internal SRAMs support the features listed hereafter, configured in RAMCFG:

- Error code correction (ECC):
  - Single error detection and correction with interrupt generation
  - Double error detection with interrupt or NMI generation
  - Status with failing address
- Write protection (1-Kbyte granularity)
- SRAM software erase

6.3 RAMCFG functional description

6.3.1 Internal SRAMs features

Five SRAMs are embedded in the devices, each with specific features:

- SRAM1, SRAM2, SRAM3 are the main SRAMs (see *Table 39*), made of several blocks that can be powered down in Stop mode to reduce consumption.
- The backup SRAM (BKPSRAM) can be retained in all low-power modes and when V_{DD} is off in VBAT mode. Refer to *Section 10: Power control (PWR)* for more details.
- SRAM2 is erased when a system reset occurs if the SRAM2_RST option bit is selected in the flash memory user option bytes. SRAM1 and SRAM3 are erased when a system reset occurs if the SRAM13_RST option bit is selected in the flash memory user option bytes. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.
- SRAM2 and (optionally) backup SRAM are protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. Refer to *Section 47: Tamper and backup registers (TAMP)* for more details.
- The RAMCFG embeds the registers related to the internal SRAMs ECC, write protection and software erase.

Table 39. SRAMs density (Kbytes)

Products	SRAM1	SRAM2	SRAM3
STM32H562/63/73xx	256	64	320
STM32H523/33xx	128	80	64

Table 40 summarizes the features supported by each internal SRAM.

Table 40. Internal SRAMs features

SRAM feature	SRAM1	SRAM2	SRAM3	BKPSRAM
Optional retention in Standby mode	-	-	-	X
Optional retention in VBAT mode	-	-	-	X
Erased with tamper detection and Backup domain reset	-	X	-	X(1)
Optionally erased with system reset	X	X	X	-
Software erase	X	X	X	X
ECC	-	X	X(2)	X
Write protection	-	X	-	-

1. Optional: BKPSRAM can be configured to be erased or not on tamper detection.
2. Available only on STM32H562/563/573xx devices.

6.3.2 Error code correction (SRAM2, SRAM3, BKPSRAM)

The ECC is supported by SRAM2, SRAM3[a] and BKPSRAM when enabled with the SRAM2_ECC, SRAM3_ECC[b] and BKPRAM_ECC user option bits. Refer to *Section 7: Embedded flash memory (FLASH)* for more details.

Seven ECC bits are added per 32 bits of SRAM, allowing two bits error detection and one bit error correction on memory read access.

As the ECC is calculated and checked for a 32-bit word, the byte and half-word write accesses are managed by the SRAM interface by first reading the whole word, then write the word again with the new byte/half-word value. ECC double errors are also detected during these byte or half-word AHB write accesses (read/modify/write done by interface). The byte or half-word write access latency is two AHB clock cycles.

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM two AHB clock cycles after the SRAM AHB is released (when SRAM is no more accessed).

Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDC/CSEDC bits are set, respectively, in the *RAMCFG memory interrupt status register (RAMCFG_MxISR)*/*RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)*. An interrupt is generated if enabled by the SEIE bit in the *RAMCFG memory x interrupt enable register (RAMCFG_MxIER)*. The failing address is stored in the *RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)* if the ALE bit is set in the *RAMCFG memory x control register (RAMCFG_MxCR)*.

Caution: Single errors cannot be detected when the SEDC bit is set.

a. Available only on STM32H562/563/573xx devices.

b. Refer to the device datasheet for the actual availability. If not present, consider this bit as reserved, and keep it at reset value.

When a double error is detected, the DED and CDED bits are set in the *RAMCFG memory interrupt status register (RAMCFG_MxISR)* and *RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)* respectively. An interrupt or NMI is generated if enabled by the DEIE or ECCNMI bit in the *RAMCFG memory x interrupt enable register (RAMCFG_MxIER)*. The failing address is stored in the *RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)* if the ALE bit is set in the *RAMCFG memory x control register (RAMCFG_MxCR)*.

Caution: Double errors cannot be detected when the DED bit is set.

SRAM3 ECC specific management

On STM32H562/563/573xx devices, when the ECC is enabled for SRAM3, only the first 256 Kbytes of SRAM3 are with ECC. The next 64 Kbytes are without ECC, and the last block is used to store the ECC, hence it cannot be used by the application.

Figure 20 shows the SRAM areas, when SRAM2 and SRAM3 ECC are enabled. *Figure 21* shows the SRAM areas, when SRAM2 ECC and SRAM3 are enabled.

Figure 20. Memory map: SRAM1, SRAM2/3 with ECC (STM32H562/72/73xx devices)

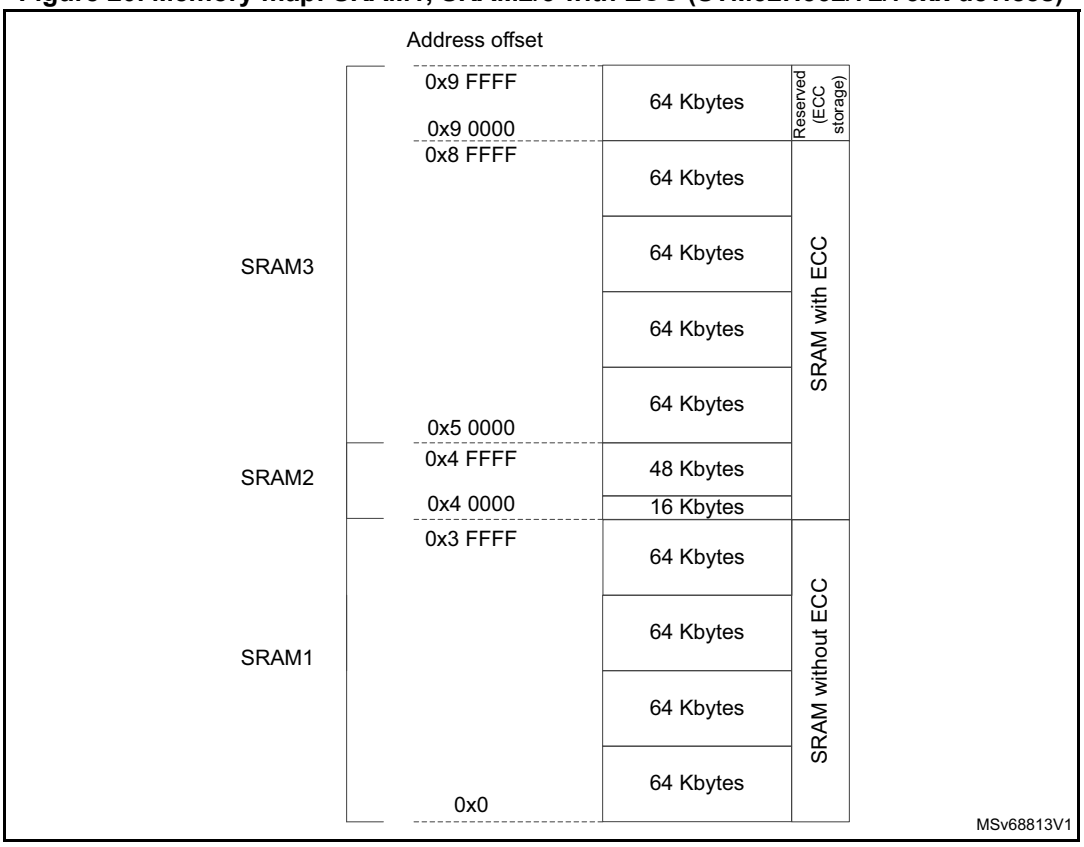
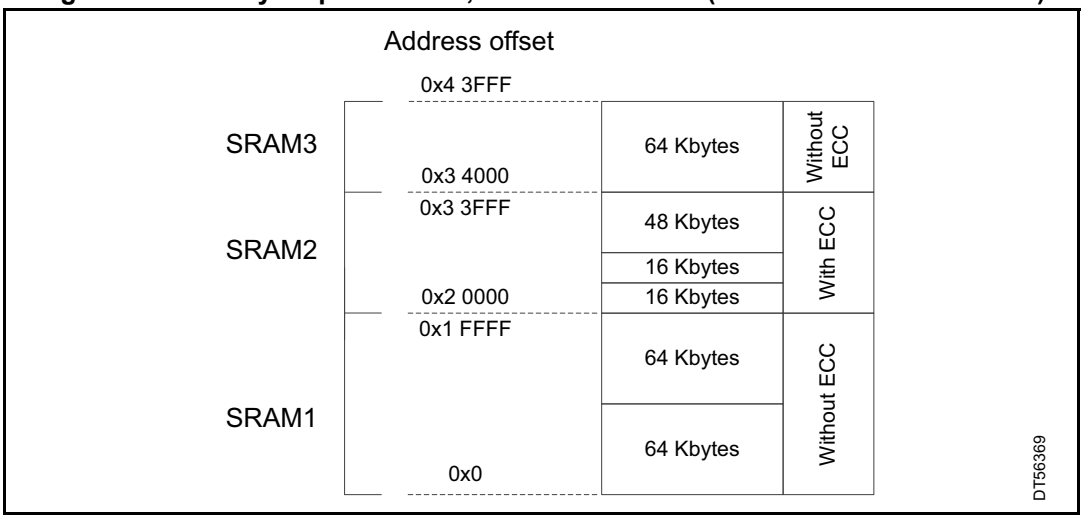


Figure 21. Memory map: SRAM1/3, SRAM2 with ECC (STM32H523/33xx devices)



When ECC is enabled by user option bits, the ECCE bit is automatically set after system reset in the related *RAMCFG memory x control register (RAMCFG_MxCR)*.

The ECC can be deactivated by executing the following software sequence:

1. Write 0xAE in the *RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)*.
2. Write 0x75 in the *RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)*.
3. Write 0 in the ECCE bit of the *RAMCFG memory x control register (RAMCFG_MxCR)*.

When ECC is deactivated (ECCE = 0), the SRAM3 ECC storage area can be read and written for ECC user test purpose. When the ECC is activated (ECCE = 1), this area is reserved for ECC storage purpose and cannot be neither read, nor written.

6.3.3 Write protection (SRAM2)

The SRAM2 is made of 1-Kbyte pages, each of them can be write-protected by setting its corresponding PxWP bit in the *RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)*, *RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)*, and *RAMCFG memory 2 write protection register 3 (RAMCFG_M2WPR3)*[a].

Table 41. SRAM2 features

Products	Page size	Number of pages	PxWP bits
STM32H562/63/73xx	1 Kbyte	64	0 to 63
STM32H523/33xx	1 Kbyte	80	0 to 79

a. This register is available only on STM32H523/533 devices. If not present, consider this register as reserved, and keep its bits at reset value.

6.3.4 Software erase

SRAM erase can be requested by executing this software sequence:

1. Write 0xCA in the *RAMCFG memory x erase key register (RAMCFG_MxERKEYR)*.
2. Write 0x53 in the *RAMCFG memory x erase key register (RAMCFG_MxERKEYR)*.
3. Write 1 in the SRAMER bit of the *RAMCFG memory x control register (RAMCFG_MxCR)*.

SRAMBUSY flag is set in the related SRAM interrupt status register as long as the erase is ongoing.

The total duration of each SRAM erase is N AHB clock cycles, where N is the size of the SRAM in 32-bit words.

If the SRAM is read or written while an erase is ongoing, wait states are inserted on the AHB bus until the end of the operation.

6.4 RAMCFG low-power modes

Table 42. Effect of low-power modes on RAMCFG

Mode	Description
Sleep	No effect. RAMCFG interrupts cause the device to exit the Sleep mode
Stop	The content of RAMCFG registers is kept
Standby	The RAMCFG peripheral is powered down and must be reinitialized after exiting Standby

6.5 RAMCFG interrupts

Table 43. RAMCFG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit(s)	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby modes
RAMCFG	ECC single error detection and correction	SEDC	SEIE	Write 1 in CSEDC	Yes	No	No
	ECC double error detection	DED	DEIE = 1 and ECCNMI = 0	Write 1 in CDED	Yes	No	No
NMI	ECC double error detection	DED	ECCNMI	Write 1 in CDED	Yes	No	No

6.6 RAMCFG registers

In the registers described below, x refers to:

- SRAM1/2/3 when x = 1/2/3, respectively
- BKPSRAM when x = 5

6.6.1 RAMCFG memory x control register (RAMCFG_MxCR)

Address offset: 0x000 + 0x040 * (x - 1), (x = 1, 2, 3, 5)

Reset value: 0x0000 000X

ECCE reset value depends on ECC enable user option bit.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in the ERASEKEY field of the RAMCFG_MxERKEYR register. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation on going

1: Erase operation on going

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: Address latch enable

0: Failing address not stored in the SRAMx ECC single/double error address registers

1: Failing address stored in the SRAMx ECC single/double error address registers

Note: For SRAM1 and SRAM3 (when ECC is not available) this bit is reserved and must be kept at reset value in control registers.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable.

This bit reset value is defined by the user option bit configuration. When set, it can be cleared by software only after writing the unlock sequence in the RAMCFG_MxECCKEYR register.

0: ECC disabled

1: ECC enabled

Note: For SRAM1 and SRAM3 (when ECC is not available) this bit is reserved and must be kept at reset value in control registers.

6.6.2 RAMCFG memory x interrupt enable register (RAMCFG_MxIER)

Address offset: 0x004 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

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

Bit 3 **ECCNMI**: Double error NMI

This bit is set by software and cleared only by a global RAMCFG reset.

0: NMI not generated in case of ECC double error

1: NMI generated in case of ECC double error

Note: if ECCNMI is set, the RAMCFG maskable interrupt is not generated whatever DEIE bit value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DEIE**: ECC double error interrupt enable

0: Double error interrupt disabled

1: Double error interrupt enabled

Bit 0 **SEIE**: ECC single error interrupt enable

0: Single error interrupt disabled

1: Single error interrupt enabled

6.6.3 RAMCFG memory interrupt status register (RAMCFG_MxISR)

Address offset: 0x008 + 0x040 * (x - 1), (x = 1, 2, 3, 5)

Reset value: 0x0000 0000

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

0: No erase operation on going

1: Erase operation on going

Note: Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option bit is enabled, tamper detection or readout protection regression. Refer to Table 40.

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

Bit 1 **DED**: ECC double error detected

0: No double error

1: Double error detected

Note: For SRAM1 and SRAM3 (when ECC is not available) this bit is reserved and must be kept at reset value in control registers.

Bit 0 **SEDC**: ECC single error detected and corrected

0: No single error

1: Single error detected and corrected

Note: For SRAM1 and SRAM3 (when ECC is not available) this bit is reserved and must be kept at reset value in control registers.

6.6.4 RAMCFG memory x ECC single error address register (RAMCFG_MxSEAR)

Address offset: 0x00C + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

Bits 31:0 **ESEA[31:0]**: ECC single error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC single error.

6.6.5 RAMCFG memory x ECC double error address register (RAMCFG_MxDEAR)

Address offset: 0x010 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

Bits 31:0 **EDEA[31:0]**: ECC double error address

When the ALE bit is set in the RAMCFG_MxCR register, this field is updated with the address corresponding to the ECC double error.

6.6.6 RAMCFG memory x interrupt clear register x (RAMCFG_MxICR)

Address offset: 0x014 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

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

Bit 1 **CDED**: Clear ECC double error detected

Writing 1 to this flag clears the DED bit in the RAMCFG_MxISR register. Reading this flag returns the DED value.

Bit 0 **CSEDC**: Clear ECC single error detected and corrected

Writing 1 to this flag clears the SEDC bit in the RAMCFG_MxISR register. Reading this flag returns the SEDC value.

6.6.7 RAMCFG memory 2 write protection register 1 (RAMCFG_M2WPR1)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 31 to 0)

These bits are set by software and cleared only by a global RAMCFG reset.

0: Write protection of SRAM2 1-Kbyte page y is disabled.

1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.8 RAMCFG memory 2 write protection register 2 (RAMCFG_M2WPR2)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 63 to 32)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.9 RAMCFG memory 2 write protection register 3 (RAMCFG_M2WPR3)

Address offset: 0x060

Reset value: 0x0000 0000

This register is available only for STM32H523/33xx devices.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P79WP	P78WP	P77WP	P76WP	P75WP	P74WP	P73WP	P72WP	P71WP	P70WP	P69WP	P68WP	P67WP	P66WP	P65WP	P64WP
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PyWP**: SRAM2 1-Kbyte page y write protection (y = 79 to 64)
These bits are set by software and cleared only by a global RAMCFG reset.
0: Write protection of SRAM2 1-Kbyte page y is disabled.
1: Write protection of SRAM2 1-Kbyte page y is enabled.

6.6.10 RAMCFG memory x ECC key register (RAMCFG_MxECCKEYR)

Address offset: 0x024 + 0x040 * (x - 1), (x = 2, 3, 5)

Reset value: 0x0000 0000

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

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

Bits 7:0 **ECCKEY[7:0]**: ECC write protection key
The following steps are required to unlock the write protection of the ECCE bit in the RAMCFG_MxCR register.
1) Write 0xAE into ECCKEY[7:0].
2) Write 0x75 into ECCKEY[7:0].
Note: Writing a wrong key reactivates the write protection.

6.6.11 RAMCFG memory x erase key register (RAMCFG_MxERKEYR)

Address offset: 0x028 + 0x040 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0000

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

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

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of the SRAMER bit in the RAMCFG_MxCR register.

1) Write 0xCA into ERASEKEY[7:0].

2) Write 0x53 into ERASEKEY[7:0].

Note: Writing a wrong key reactivates the write protection.

6.6.12 RAMCFG register map

Table 44. RAMCFG register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**RAMCFG_M1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x04	Reserved	Reserved																															
0x08	**RAMCFG_M1ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x0C to 0x24	Reserved	Reserved																															
0x28	**RAMCFG_ M1ERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x2C to 0x3C	Reserved	Reserved																															
0x40	**RAMCFG_M2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x44	**RAMCFG_M2IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x048	**RAMCFG_M2ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x04C	**RAMCFG_ M2SEAR**	ESEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	**RAMCFG_ M2DEAR**	EDEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	**RAMCFG_M2ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x058	**RAMCFG_ M2WPR1**	P31WP	P30WP	P29WP	P28WP	P27WP	P26WP	P25WP	P24WP	P23WP	P22WP	P21WP	P20WP	P19WP	P18WP	P17WP	P16WP	P15WP	P14WP	P13WP	P12WP	P11WP	P10WP	P9WP	P8WP	P7WP	P6WP	P5WP	P4WP	P3WP	P2WP	P1WP	P0WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	**RAMCFG_ M2WPR2**	P63WP	P62WP	P61WP	P60WP	P59WP	P58WP	P57WP	P56WP	P55WP	P54WP	P53WP	P52WP	P51WP	P50WP	P49WP	P48WP	P47WP	P46WP	P45WP	P44WP	P43WP	P42WP	P41WP	P40WP	P39WP	P38WP	P37WP	P36WP	P35WP	P34WP	P33WP	P32WP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	**RAMCFG M2WPR3(1)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	P79WP	P78WP	P77WP	P76WP	P75WP	P74WP	P73WP	P72WP	P71WP	P70WP	P69WP	P68WP	P67WP	P66WP	P65WP	P64WP
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	**RAMCFG_ M2ECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 44. RAMCFG register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x068	**RAMCFG_ M2ERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x06C to 0x07C	Reserved	Reserved																															
0x080	**RAMCFG_M3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				0
0x084	**RAMCFG_M3IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x088	**RAMCFG_M3ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x08C	**RAMCFG_ M3SEAR**	ESEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	**RAMCFG_ M3DEAR**	EDEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	**RAMCFG_M3ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x098 to 0x0A0	Reserved	Reserved																															
0x0A4	**RAMCFG_ M3ECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0A8	**RAMCFG_ M3ERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0AC to 0x0FC	Reserved	Reserved																															
0x100	**RAMCFG_M5CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	ALE	Res.	Res.	Res.	ECCE
	Reset value																								0				0				x
0x104	**RAMCFG_M5IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI	Res.	DEIE	SEIE
	Reset value																													0		0	0
0x108	**RAMCFG_M5ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x10C	**RAMCFG_ M5SEAR**	ESEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110	**RAMCFG_ M5DEAR**	EDEA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 44. RAMCFG register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x114	**RAMCFG_M5ICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x118 to 0x120	Reserved	Reserved																															
0x124	**RAMCFG_ M5ECCKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x128	**RAMCFG_ M5ERKEYR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

1. Available only on STM32H523/33xx devices.

Refer to *Section 2.3* for the register boundary addresses.

7 Embedded flash memory (FLASH)

7.1 Introduction

The embedded flash memory (FLASH) manages the accesses of any master to the up to 2 Mbytes of embedded nonvolatile memory. It implements read, program, and erase operations, error corrections, as well as various integrity and confidentiality protection mechanisms.

FLASH manages the automatic loading of nonvolatile user option bytes at power-on reset, and implements the dynamic update of these options. It also features a high-cycle data area, a one-time-programmable (OTP) area, a secure key storage area (OBKeys), and a read-only area configured by STMicroelectronics during manufacturing.

7.2 FLASH main features

- Up to 2 Mbytes of nonvolatile memory, divided into two 1-Mbyte banks
- Flash memory read operations supporting multiple lengths: 128 bits, 64 bits, 32 bits, 16 bits, or one byte
- Flash memory programming by 128 (user area, OBKeys) and by 16 bits (OTP and flash high-cycle data area)
- 8-Kbyte sector erase, bank erase and dual-bank mass erase
- Dual-bank organization supporting:
  - Simultaneous operations: read-while-write (program and erase) is supported, including flash high-cycle data area. The two banks share the same interface, hence write and erase cannot be performed in parallel (write-while-write is not supported).
  - Bank swapping: the address mapping of the user memory of each bank can be swapped, along with the corresponding registers. Security flags remain valid for the physical bank, so the data are not reveled by swapping to a bank with lower security configuration.
- Error code correction (ECC): one error detection/correction, or two errors detection per 128-bit flash word using nine ECC bits, on 16-bit words with six bits within configurable flash high-cycle data area.
- User configurable nonvolatile option bytes
- Flash memory enhanced protections, activated by option bytes
  - different product states for protecting memory content from debug access
  - sector group write-protection (WRPSG), protecting up to 32 groups of four sectors (32 Kbytes) per bank
  - two secure-only areas (one per user flash bank): when enabled, these areas are accessible only if the microcontroller operates in Secure access mode
  - HDP protection, providing temporal isolation for startup code
- 2-Kbyte one-time programmable (OTP) area
- Read-only area configured by STMicroelectronics
- Prefetch reads the next sequential instruction from flash memory

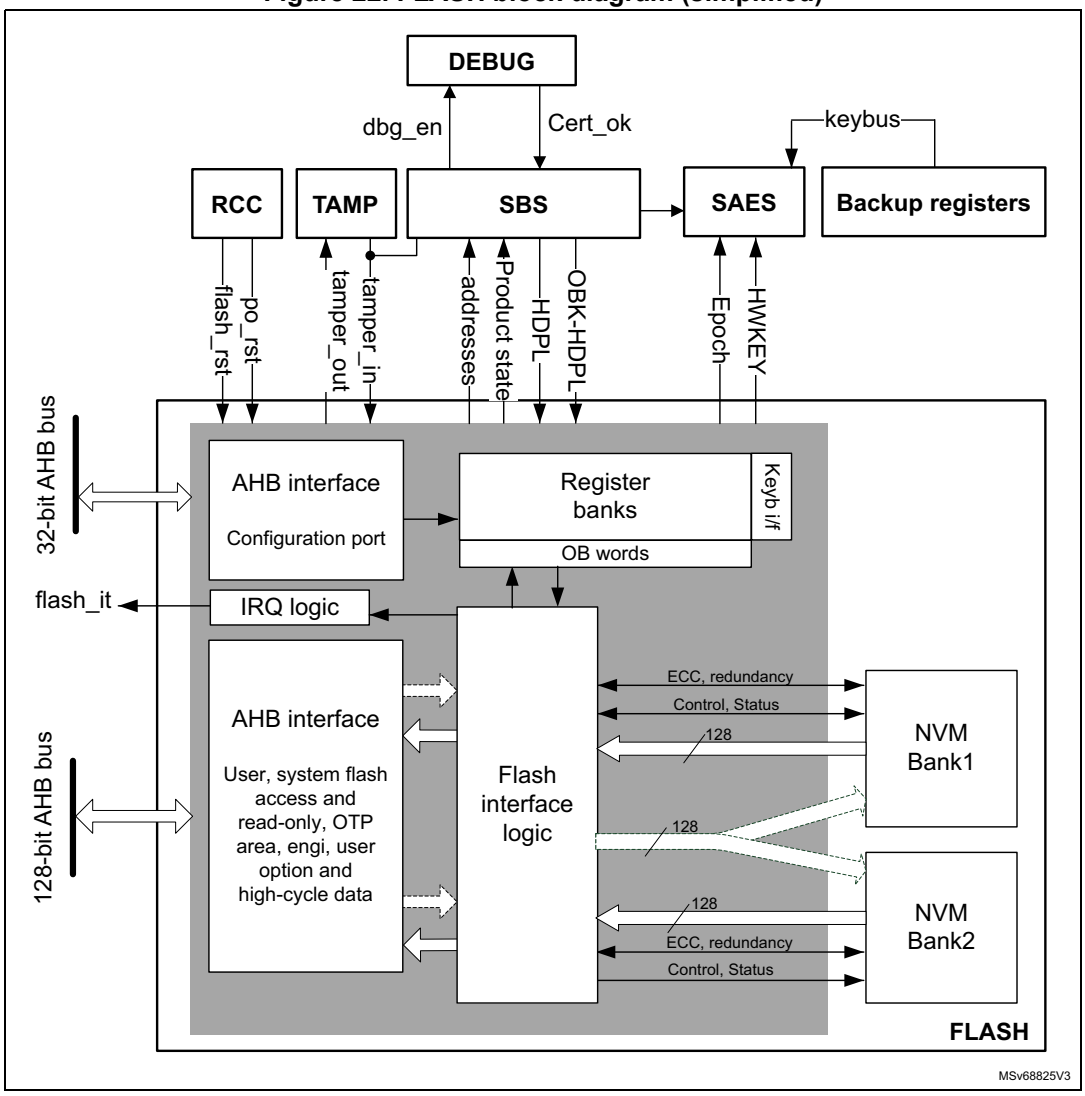
- Up to 48 Kbytes per bank supporting high-cycling capability (100 kcycles), to be used for data (EEPROM emulation)

7.3 FLASH functional description

7.3.1 FLASH block diagram

Figure 22 shows the embedded flash memory block diagram.

Figure 22. FLASH block diagram (simplified)



7.3.2 FLASH signals

The flash memory has two AHB connections, namely the flash AHB register interface and the main AHB interface.

Flash AHB register interface

- Data size is 32 bits
- Except for some registers (FLASH_NS/SECKEYR, FLASH_NS/SECOBKKEYR and FLASH_OPTKEYR, used to insert unlock sequences for control, and option registers that can be written by 32 bits), it is possible to read and write all registers by 8, 16 and 32 bits.
- When unlock sequence for control and option registers is wrong, a bus error is raised, otherwise no read or write errors are generated on the bus.

Main AHB interface

The AHB data bus size is 128 bits. This interface is used to handle three different targets:

- Code placed in user and system memory, protected by 9 bits of ECC.
- Secure keys placed in OBKey sectors, protected by 9 bits of ECC.
- OTP, read-only and flash memory high-cycle data area. protected by 6 bits of ECC.

The main AHB interface is implemented as follows:

- User and system memory, OBKeys storage:
  - Supports multiple length: 128-, 64-, 32-, 16- and 8-bit data width.
  - There is a read buffer of 128 bits for each bank where the last data read is stored. If data are available in the read buffer, no read access is given to the flash memory. Buffer is flushed when write access, OTP access, OBK swap, OBK alternate sector erase, high-cycle data area access, user option change request or erase operation occur.
  - There is a prefetch of the same size as the read buffer.
  - A 9-bit ECC is associated to each 128-bit data flash memory word.
- OTP, read-only and flash high-cycle data
  - Two dedicated data buffer of 137 bits are used to manage 16-bit data with 6-bit ECC
  - Reading two times the same address triggers two flash read accesses.
  - During a read access, two wait states are added in addition to the memory wait states. These wait states are necessary to parse the data buffer.
  - Each write access triggers a write in the flash memory.
  - 6-bit ECC is associated to each 16-bit data.

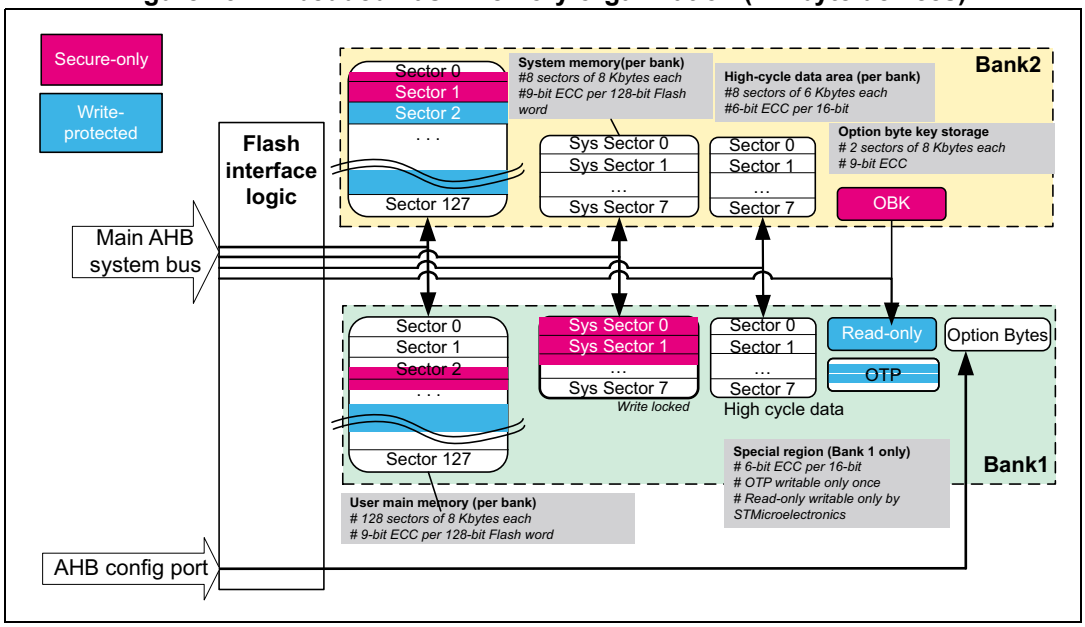
By default, all the AHB memory range is cacheable. For regions where caching is not practical (OTP, RO, data area), MPU must be used to disable local cacheability.

7.3.3 Flash memory architecture and usage

Flash memory architecture

Figure 23 shows the organization supported by the embedded flash memory. By default, all the AHB memory range is cacheable. For regions where caching is unpratical (OTP, read-only, data) use MPU to disable local cacheability.

Figure 23. Embedded flash memory organization (2-Mbyte devices)



The embedded flash nonvolatile memory is composed of:

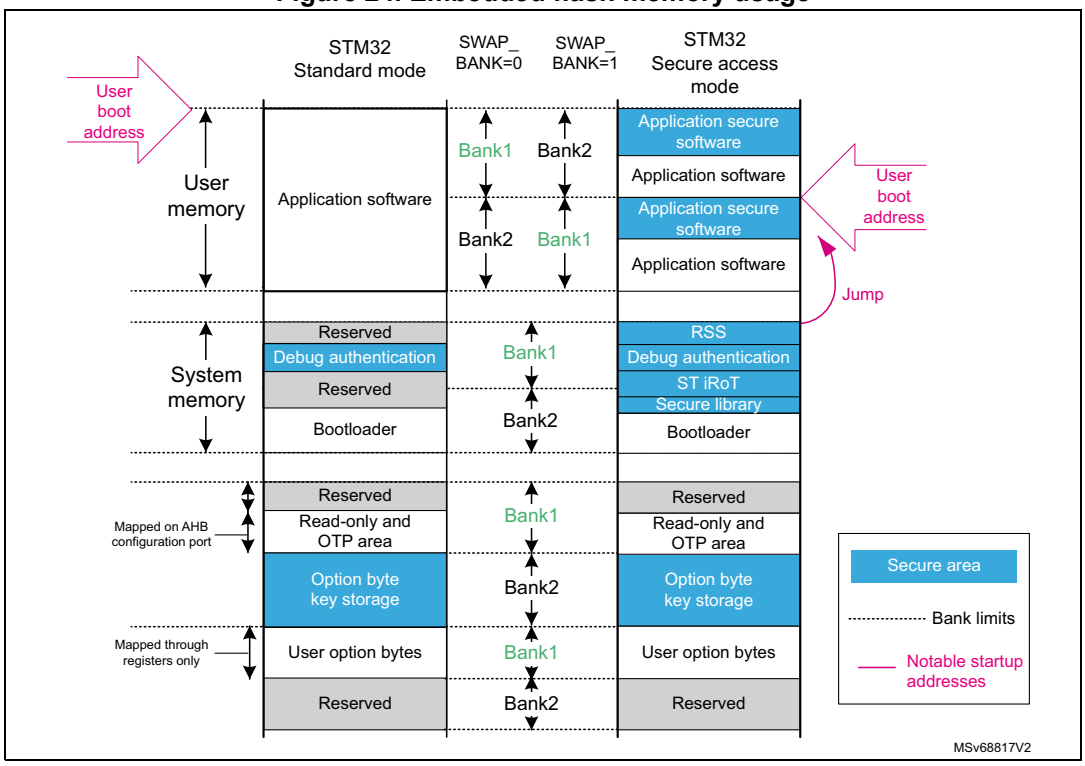
- A main memory block, organized in two banks. Each bank is divided in sectors of 8 Kbytes each, and features flash-word rows of 128 bits + 9 bits of ECC per word (see *Table 48*).
- A system memory block of 128 Kbytes, divided into two 64-Kbyte banks. Each bank is divided in eight 8-Kbyte sectors. The system flash memory is ECC protected (9-bit ECC per 128-bit word).
- A set of nonvolatile option bytes loaded at reset by the embedded flash memory and accessible by the application software only through the AHB configuration register interface.
- A 2-Kbyte one-time-programmable (OTP) area that can be written only once by the application software.
- A 2-Kbyte read-only area. It contains a unique device ID and product information.
- Two memory sectors (2 x 8 Kbytes) of secure key storage, OBKeys.
- Up to 16 sectors of user flash memory with high cycling capability (100 K cycles) for data, 8 sectors per bank.

The overall flash memory architecture and its corresponding access interface is summarized in *Table 48*.

Partition usage

Figure 24 shows how the embedded flash memory is used by STMicroelectronics and by the application software.

Figure 24. Embedded flash memory usage



User and system memories are used differently, according to product state and other option bytes settings:

- The user memory contains the application code and data, while the system memory is used with root secure services (RSS), the debug authentication code, and the STM32 bootloader. When a reset occurs, the core jumps to the boot address configured through the BOOT pin, the (SEC/NS)BOOTADD option bytes, and the product state.
- The unique boot entry (BOOT_UBE) is set to either ST iRoT located in the system flash memory, or iRoT in the user flash memory. This feature is available only on cryptography enabled devices.
- Secure service library in system flash memory is available in secure mode.
- If the debugger is attached to the product, the entry point is the debug authentication policy, used to unlock the device via the SBS when attached to debugger. A digital signature must be provided to perform a regression to product state, where debug is allowed.
- If (SEC/NS)BOOTADD is not yet configured, dedicated libraries can be used for secure boot. They are located in the system flash memory:
  - ST libraries in system flash memory assist the application software boot with special features, such as secure boot and secure firmware install (SFI-RSS)
  - ST iROT (immutable root of trust) secure software in user flash memory is used for secure firmware update and provisioning (SFU)

Note: For more information on option byte setup for boot, refer to Section 7.4.6.

7.3.4 FLASH read operations

Read operation overview

Read access to main and system flash memory operates as follows:

- There is a 128-bit read data buffer associated to each bank, which stores the last data read. If several consecutive read accesses request data belonging to the same flash data word (128 bits), data are read directly from the current data read buffer, without triggering additional flash read operations. This mechanism occurs each time a read access is granted. When a read access is rejected for security reasons, the corresponding read error response is issued by the embedded flash memory and no read operation to flash memory is triggered.
- The read data buffer is disabled when write access or OTP access, user option change request, OBK swap, OBK erase or other erase operation occur.

Read access to OTP, RO and flash high-cycle data operates as follows:

1. Flash data word of 137 bits is read and stored in a temporary buffer.
2. The interface parses the 137-bit data word, and selects the 16- or 32-bit data requested.
3. While parsing the 137-bit data word, (two) wait states are added, and the AHB bus is stalled.
4. If the application reads an OTP data or flash high-cycle data not previously written, **a double ECC error** is reported and **only a word full of set bits is returned** (see *Section 7.3.9* for details). The read data (in 16 bits) is stored in FLASH_ECCDR register, so that the user can identify if the double ECC error is due to a virgin data or a real ECC error.
5. Reading two times the same address triggers two reads in the flash memory.
6. For 8-bit accesses, an AHB bus error is generated.

Note: The embedded flash memory can perform single-error correction and double-error detection while read operations are being executed (see Section 7.3.8).

Instruction prefetch

The Cortex-M33 fetches instructions and literal pools (constants/data) over the C-Bus and through the instruction cache, if it is enabled. The prefetch block aims at increasing the efficiency of C-Bus accesses when the instruction cache is enabled, by reducing the cache refill latency.

Prefetch is efficient in case of sequential code; prefetch in the flash memory allows the next sequential instruction line to be read from the memory, while the current instruction line is being filled in instruction cache and executed by the CPU.

Prefetch is enabled by setting the PRFTEN bit in the FLASH access control register (FLASH_ACR). PRFTEN must be set only if at least one wait state is needed to access the flash memory.

Adjusting read timing constraints

The embedded clock must be enabled and running before reading data from a nonvolatile memory.

To correctly read data from the memory, the number of wait states (LATENCY) must be correctly programmed in the access control register (FLASH_ACR), according to the main AHB interface clock frequency, and the internal voltage range of the device (V_{core}).

Table 45 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded flash memory clock frequency, and the supply voltage range.

Table 45. Recommended number of wait states and programming delay

Number of wait states (LATENCY)	Programming delay (WRHIGHFREQ)	Interface clock frequency vs. V_{CORE} range(1)			
		VOS3 range 0.95 to 1.05 V	VOS2 range 1.05 to 1.15 V	VOS1 range 1.15 to 1.26 V	VOS0 range 1.30 to 1.40 V
0 WS (1 FLASH clock cycle)	00	0 to 20 MHz	0 to 30 MHz	0 to 34 MHz	0 to 42 MHz
1 WS (2 FLASH clock cycles)		20 to 40 MHz	30 to 60 MHz	34 to 68 MHz	42 to 84 MHz
2 WS (3 FLASH clock cycles)	01	40 to 60 MHz	60 to 90 MHz	68 to 102 MHz	84 to 126 MHz
3 WS (4 FLASH clock cycles)		60 to 80 MHz	90 to 120 MHz	102 to 136 MHz	126 to 168 MHz
4 WS (5 FLASH clock cycles)	10	80 to 100 MHz	120 to 150 MHz	136 to 170 MHz	168 to 210 MHz
5 WS (6 FLASH clock cycles)		N/A	N/A	170 to 200 MHz	210 to 250 MHz

1. Voltage range from 1.26 to 1.30 V is not supported.

Adjusting system frequency

After power-on, the embedded flash memory is clocked by the 64 MHz high-speed internal oscillator (HSI), with a voltage range set at a scaled value of VOS3: a conservative 3 wait-state latency is specified in FLASH_ACR register (see *Table 45*).

When changing the bus frequency, the application software must follow the sequence described below, to tune the number of wait states required to access the memory.

To increase the CPU frequency:

1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in *Table 45.*
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded flash memory clock source and/or the new AHB clock prescaler value are taken in account by reading back the embedded flash memory clock source status and/or the prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the CPU frequency:

1. Modify the embedded flash memory clock source and/or the clock prescaler in the RCC_CFGR register of reset and clock controller (RCC).
2. Check that the embedded flash memory new clock source and/or the new clock prescaler value are taken into account by reading back the embedded flash memory clock source status and/or the AHB interface prescaler value in the RCC_CFGR register of reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in FLASH_ACR register, as described in *Table 45.*
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

Error code correction (ECC)

The memory embeds an error correction mechanism. Single-error correction and double-error detection are performed for each read operation. For more details, refer to *Section 7.3.8*.

Read errors

When the ECC mechanism is unable to correct the read operation, the memory reports read errors, as described in *Section 7.9.10*.

Read interrupts

See *Section 7.10* for details.

7.3.5 FLASH program operations

Program operation overview

Program operation consists in issuing write commands. The memory supports the execution of only one write-command at a time. Write-while-write is not supported. Nothing prevents overwriting a non-virgin flash word, but this is not recommended. The result may lead to invalid data and inconsistent ECC code.

User flash, OBK storage and system flash memories sectors

For the user and system flash memories, 9-bit ECC is associated to each 128-bit data flash word. In this case, the embedded flash memory must always perform write operations to nonvolatile memory with a 128-bit word granularity. Once the write buffer is full (128 bits), the Busy flag is set, and a programming operation is triggered.

There is a write buffer common to Bank1 and 2, which supports multiple write-access types (128, 64, 32, 16 or 8 bits). The application can decide to write from 8 bits to 128 words. In this case, a force-write mechanism to the 128 bits + ECC is used (see FW bit of FLASH_NS/SECCR register).

When the write request is issued to the memory, any new write request stalls the main AHB bus. Moreover, while a write operation is ongoing, any new read request to the same bank stalls the main AHB bus.

OTP, RO, flash high-cycle data

When the target memory is OTP, RO and flash high-cycle data sectors, 6-bits ECC code is associated to each 16-bit data flash word. The embedded flash memory supports 16- or 32-bit write operations (8-bit write operations are not supported).For 8-bit accesses, write accesses are ignored. There is no write data buffer. Each write access triggers a write in the flash memory.

Note: The OTP area is typically write-protected on the final product, as described in Section 7.3.9.

The write protection check is performed at the reception of the write request (during address phase).Write protection is not checked anymore at the output of the write buffer. If a write protection violation is detected, the write operation is canceled, and write protection error (WRPERR) is raised in FLASH_NS/SECSR register.

Note: For write protections of main flash, ICP, and OTP see Section 7.6.

Monitoring ongoing write operations

The application software can use a status flag located in FLASH_NS/SECSR to monitor ongoing write operations. Since only one operation is possible at a time, this flag indicates if any operation (write, erase, option change) is ongoing, whatever the bank.

- **BSY**: indicates that an effective write, erase, option byte change, OBK swap, OBK alt sector erase is ongoing in the nonvolatile memory. This flag is not dedicated to a specific bank. It is set when an operation is starting on the memory, whatever the bank. An operation is triggered by:
  - An erase (FLASH_NS/SECCR.STRT)
  - A write (FLASH_NS/SECCR.PG + AHB write)
  - An option modification (FLASH_OPTCR.OPTSTRT)
  - OBKeys sector swap and OBKeys sector erase (FLASH_NS/SECOBKCFGR.SWAP_SECT_REQ and FLASH_NS/SECOBKCFGR.ALT_SECT_ERASE)

They are cleared when the current operation ends, or in case of error.

- **WBNE**: this bit indicates that the embedded flash memory is waiting for new data to complete the 128-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, or forces the writes by using FW bit in FLASH_NS/SECCR, or an error is detected. When WBNE is high, it is not possible to launch an erase, an option modification, an OBK swap or OBK alternate sector erase operation on flash memory.
- **DBNE**: this bit indicates that the data buffer for parsing 16-bits data is not empty:
  - 16-bit data write access is received, and the data buffer is being filled. It is set at the receipt of a valid write access, and reset as soon as the write request preparation has been processed.

Note: If the memory is busy at the receipt of the AHB write request, the CPU execution is stalled.

Enabling write operations

Before programming the user flash memory in Bank1 or in Bank2, the application software must ensure that the PG bit is set to 1 in FLASH_NS/SECCR. If not, an unlock sequence must be used (see *Section 7.6.7*), and the PG bit must be set.

When an option byte or an option byte key must be modified, or a mass erase must be started, the application software must ensure that FLASH_OPTCR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 7.6.7*).

A separate mechanism with similar use exists for FLASH_NS/SECOBKCFGR. The control register must be unlocked prior to the start of any OBKeys storage modification. Writing the correct sequence to the FLASH_NSOBKKEYR unlocks FLASH_NSOBKCFGR. FLASH_SECOBKKEYR is linked to FLASH_SECOBKCFGR, as described in *Section 7.6.7*.

Note: The application software must not unlock an already unlocked register, otherwise this register remains locked until the next system reset.

If needed, the application software can update the programming delay, as described in *Adjusting programming timing constraints*.

Writing to the FLASH control register FLASH_NS/SECCR and FLASH_OPTCR

The FLASH_NS/SECCR, FLASH_OPTCR and FLASH_NS/SECOBKCFGR registers are not accessible in write mode when the BSY bit is set. Any attempt to write these registers while the BSY bits is set causes the AHB bus to stall until SEC/NSBSY bit is cleared.

Single-write sequence

The recommended single-write sequence is the following:

1. Make sure protection mechanism does not prevent programming
2. Check that no memory operations are ongoing by checking the BSY bit in the FLASH_NS/SECSR register and CDBNE bits in the FLASH_NS/SECSR register. Check that the write buffer is empty by checking the WBNE bit in the FLASH_NS/SECSR register
3. Check and clear all the error flags due to previous programming/erase operation
4. Unlock the FLASH_NS/SECCR register, as described in *Section 7.6.7* (only if this register is not already unlocked)
5. Enable write operations by setting PG bit in the FLASH_NS/SECCR register
6. Write one flash-word at aligned address

Note: NS/SECWBNE flag indicates if the 128-bit write buffer is waiting for new data.

Note: No erase request, options change request, OBK operation is allowed between the first write and the completion of the write operation.

7. Wait for the BSY bit to be cleared in the corresponding FLASH_NS/SECSR register
8. Clear PG bit in FLASH_NS/SECCR register if there are not any more programming requests

If step *6* is executed incrementally (for example byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using FW bit in FLASH_NS/SECCR register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.

Note: The usage of a force-write operation prevents the application from updating, in a later stage, the missing bits with a value different from 1. This can lead to unexpected or inconsistent data, or ECC.

Adjusting programming timing constraints

Program operation timing constraints depend of the memory clock frequency, which directly impacts the performance. If timing constraints are too tight, the nonvolatile memory does not operate correctly, if they are too lax, the programming speed is not optimal.

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to *Table 45* for the recommended programming delay, depending upon the memory clock frequency.

FLASH_ACR configuration register is common to both banks.

The application software must check that no program/erase operation is ongoing before modifying WRHIGHFREQ.

Caution: Modifying WRHIGHFREQ while programming/erasing the memory can corrupt its content.

Programming errors

When a program operation fails, an error is reported, as described in *Section 7.9*.

Programming interrupts

See *Section 7.10: FLASH interrupts* for details.

7.3.6 FLASH erase operations

Erase operation overview

The memory can perform erase operations on 8-Kbyte user sectors, on one user flash memory bank, or on two user flash memory banks (for example mass erase). For more details in user flash memory, ICP, user options and OTP erase protection, see *Section 7.6*.

Erase commands are issued through the AHB configuration interface. The memory supports one operation at a time, if it receives simultaneously a write and an erase request an error flag is raised, and both operations are canceled. See *Section 7.9* for details.

After successful completion of erase, all the bytes in the erased flash sectors are set to the flash default 0xFF value.

Erase and WRP

If the application software attempts to erase a write-protected user sector, the sector erase operation is aborted, and the WRPERR flag is raised in the FLASH_NS/SECSR register, as described in *Section 7.9.2*.

Flash busy

Busy signals is described in *Monitoring ongoing write operations*.

Writing to the FLASH control register FLASH_NS/SECCR and FLASH_OPTCR

Refer to *Writing to the FLASH control register FLASH_NS/SECCR and FLASH_OPTCR*.

Enabling erase operations

Before erasing a sector, the application software must make sure that FLASH_NS/SECCR is unlocked. If this is not the case, an unlock sequence must be used (see *Section 7.6.7*).

Note: The application software must not unlock a register that is already unlocked, otherwise this register remains locked until next system reset. This can be used to deliberately lock-out a register from further accesses.

Similar constraints apply to bank erase requests.

Standard flash sector erase sequence

To erase an 8- or 6-Kbyte data user sector without security protections, proceed as follows:

1. Make sure protection mechanism does not prevent sector erase (WRP, secure flag, HDP).
2. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register, and that the write buffer is empty by checking the WBNE bit in the FLASH_NSSR register.
3. Check and clear all the non-secure error flags due to previous programming/erase operation. Refer to *Section 7.9* for details.
4. Unlock the FLASH_NSCR register, as described in *Section 7.6.7* (only if this register is not already unlocked).
5. Set the BKSEL bit, the SER bit and SNB bitfield in the FLASH_NSCR register. BKSEL indicates in which physical bank sector must be erased, then SER indicates a sector erase operation, while SNB contains the target sector number. In case of data sector, use the number of the corresponding regular sector.
6. Set the STRT bit in the FLASH_NSCR register.
7. Wait for the BSY bit to be cleared in the FLASH_NSSR register.
8. STRT bit is automatically cleared at the end of the sector erase, or in case of error.
9. Clear SER in FLASH_NSCR register if there are not anymore sector erase request to be issued.

Note: If another erase flag is requested simultaneously to the sector erase, a PGSERR error is generated.

Secure flash memory sector erase sequence

To erase an 8- or 6-Kbyte data secure configured user sector, proceed as follows:

1. Make sure write protection or HDP mechanism does not prevent sector erase.
2. Ensure that no memory operations are ongoing (by checking the BSY and DBNE bits in the FLASH_SECSR register), and that the write buffer is empty (by checking the WBNE bit in the FLASH_SECSR register).
3. Check and clear all the secure error flags due to previous programming/erase operations. Refer to *Section 7.9* for details.
4. Unlock the FLASH_SECCR register, as described in *Section 7.6.7* (only if this register is not already unlocked).
5. Set the BKSEL bit, the SER bit and SNB bitfield in the FLASH_SECCR register. BKSEL indicates in which physical bank sector must be erased, then SER indicates a sector erase operation, while SNB contains the target secure sector number. In case of data sector, use the number of the corresponding regular sector.

6. Set the STRT bit in the FLASH_SECCR register.
7. Wait for the BSY bit to be cleared in the FLASH_SECSR register.
8. STRT bit is automatically cleared at the end of the sector erase or in case of error.
9. Clear SER in FLASH_SECCR register if there are not any secure sector erase request to be issued.

Note: If another erase flag is requested simultaneously to the sector erase, a PGSERR error is generated.

Standard flash memory bank erase sequence

To erase bank where no sector is configured as secure:

1. Make sure protection mechanism does not prevent sector erase.
2. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the non-secure error flags due to previous programming/erase operation. Refer to *Section 7.9* for details.
4. Unlock the FLASH_NSCR register, as described in *Section 7.6.7* (only if this register is not already unlocked).
5. Set the BKSEL bit and the BER bit in the FLASH_NSCR register to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in the FLASH_NSCR register to start the bank erase operation. Then wait until the BSY bit is cleared in the FLASH_NSSR register.
7. STRT bit is automatically cleared at the end of erase sequence or in case of error.
8. Clear BER in FLASH_NSCR register if there is not other bank erase request to be issued.

Secure flash memory bank erase sequence

To erase bank where all sectors are configured with secure flag:

1. Make sure protection mechanism does not prevent sector erase (for mass erase the HDP is also considered, it cannot be fully executed from HDPL= 2, 3 if HDP is defined).
2. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_SECSR register and that the write buffer is empty by checking the WBNE bit in the same register.
3. Check and clear all the secure error flags due to previous programming/erase operation. Refer to *Section 7.9* for details.
4. Unlock the FLASH_SECCR register, as described in *Section 7.6.7* (only if this register is not already unlocked).
5. Set the BKSEL bit and the BER bit in the FLASH_SECCR register to the targeted physical bank (swap setting is ignored).
6. Set the STRT bit in the FLASH_SECCR register to start the bank erase operation. Then wait until the BSY bit is cleared in the FLASH_SECSR register.
7. STRT bit is automatically cleared at the end of erase sequence or in case of error.
8. Clear BER in FLASH_SECCR register if there is not other bank erase request to be issued.

Flash mass erase sequence

To erase all sectors of both banks, using non-secure access, all sectors must be configured as non-secure. The application software can set the MER bit to 1 in FLASH_NSCR register, as described below:

1. Make sure protection mechanisms do not prevent mass erase (*Section 7.6*).
2. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NSSR register.
3. Check and clear all the non-secure error flags due to previous programming/erase operation. Refer to *Section 7.9* for details.
4. Unlock the FLASH_NSCR register as described in *Section 7.6.7* (only if the registers are not already unlocked).
5. Set the MER bit to 1 in FLASH_NSCR register.
6. Set the STRT bit in the FLASH_NSCR register. Then wait until BSY bit is cleared in the FLASH_NSSR register.
7. STRT bit is cleared automatically at the end of the erase sequence, or in case of error.
8. Clear MER in FLASH_NSCR register.

Secure flash memory mass erase sequence

To erase all sectors of both banks simultaneously, using secure access, all sectors must be configured as secure. The application software can set the MER bit to 1 in FLASH_SECCR register, as described below:

1. Make sure protection mechanisms do not prevent mass erase (*Section 7.6*).
2. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SECSR register.
3. Check and clear all the secure error flags due to previous programming/erase operation. Refer to *Section 7.9* for details.
4. Unlock the FLASH_SECCR register as described in *Section 7.6.7* (only if the registers are not already unlocked).
5. Set the MER bit to 1 in FLASH_SECCR register.
6. Set the STRT bit in the FLASH_SECCR register. Then wait until BSY bit is cleared in the FLASH_SECSR register.
7. STRT bit is cleared automatically at the end of the erase sequence, or in case of error.
8. Clear MER in FLASH_SECCR register.

Note: Mass and bank erase also erase high-cycle data sectors aliased from the erased bank.

7.3.7 FLASH parallel operations

As the memory is divided into two independent banks, the embedded flash memory interface supports a read in one bank while a write (RWW. read while write) or an erase is executed in the other bank. It does not support write-while-write, nor read-while-read. Same is valid for the high-cycle data area, system flash libraries, or the OBK (located on Bank2).

In all cases, the sequences described in *Section 7.3.4*, *Section 7.3.5* and *Section 7.3.6* apply.

7.3.8 FLASH error protections

Error correction codes (ECC)

The embedded flash memory supports an error correction code (ECC) mechanism, based on the SECDED algorithm, to correct single errors and detect double errors.

This mechanism uses nine ECC bits per 128-bit flash word, and applies to user and system memory.For read-only, OTP, flash high-cycle data, a stronger six ECC bits per 16-bit word is used. A double ECC error is generated for an OTP or flash high-cycle data virgin word (for example a word with 22 bits at 1). When this OTP or flash high-cycle data word is no more virgin, the ECC error disappears.

More specifically, during each read operation from a 128-bit flash word, the embedded flash memory retrieves the 9-bit ECC information, computes the ECC of the flash word, and compares the result with the reference value. If they do not match, the corresponding ECC error is raised, as described in *Section 7.9.10*.

During each program operation, a 9-bit ECC code is associated to each 128-bit data flash word, and the resulting 137-bit flash word information is written in nonvolatile memory.

A similar mechanism applies to read-only and OTP areas, but with 6-bit ECC for 16-bit data.

7.3.9 OTP and RO memory access

OTP and RO memory are accessed through main AHB interface. The OTP is accessible at addresses 0x08FF_F000 to 0x08FF F7FF, and the read-only section is accessible from 0x08FF F800 to 0x08FF_FFFF.

FLASH one-time programmable area

The embedded flash memory offers a 2048-byte memory area dedicated to application non-confidential, one-time programmable data (OTP). This area is composed by 1024 words of 16 bits (plus 6 bits of ECC). It cannot be erased, and can be written only once. The OTP area can be accessed through the main AHB interface from address 0x08FF F000 to 0x08FF F7FE.

OTP data can be programmed by the application software by 16-bit chunks. Overwriting an already programmed16-bit half-word can lead to data and ECC errors, and is therefore not supported.

Note: The OTP area is virgin when the device is delivered by STMicroelectronics.

When reading OTP data with a single error corrected or a double error detected, the embedded flash memory reports read errors, as described in *Section 7.9.10*.

When reading OTP data not written by the application software (such as virgin OTP), the ECC correction reports a double-error detection (ECCD), and the data are to be found in the FLASH_ECCDR register. ECCD implies an NMI raised.

OTP write protection

OTP data are organized as 32 blocks of 32 OTP words, as shown in *Table 46*. An entire OTP block can be protected (locked) from write accesses by setting the LOCKBLi bit (i = 0 to 31) corresponding to each OTP block in the FLASH_OTPBLR register. A block can be write-protected, if it has been programmed (even partially) or not.

The OTP block locking operation is irreversible, and independent from the product life state.

Note: The OTP area can be accessed only in read mode.

Table 46. Flash memory OTP organization

OTP block	AHB address	AHB word		Lock bit
		[31:16]	[15:0]	
Block 0	0x08FF F000	OTP001	OTP000	LOCKBL0
	0x08FF F004	OTP003	OTP002	
	...			
	0x08FF F03C	OTP031	OTP030	
Block 1	0x08FF F040	OTP033	OTP032	LOCKBL1
	0x08FF F044	OTP035	OTP034	
	...			
	0x08FF F07C	OTP063	OTP062	
Block 2	0x08FF F080	OTP065	OTP064	LOCKBL2
	0x08FF F084	OTP067	OTP066	
	...			
	0x08FF F0BC	OTP95	OTP94	
...				
Block 31	0x08FF F7C0	OTP993	OTP992	LOCKBL31
	0x08FF F7C4	OTP995	OTP994	
	...			
	0x08FF F7FC	OTP1023	OTP1022	

OTP write sequence

Follow the sequence below to write an OTP word:

1. Check that no memory operations are ongoing by checking the BSY bit in the FLASH_NSSR register and that the data buffer is empty by checking the DBNE bit in the FLASH_NSSR register.
2. Check and clear all the error flags due to previous programming/erase operation.
3. Set PG bit in the FLASH_NSCR register.
4. Check the protection status of the target OTP word (see *Table 46*). The corresponding LOCKBLi bit must not be set to 1.
5. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in *Table 46*. Alternatively, the application software can program separately the 16-bit MSB or 16-bit LSB. In this case the first 16-bit write operation starts immediately, without waiting for the second one.
6. Wait for the BSY bit to be cleared in the FLASH_NSSR register.
7. Clear PG bit in FLASH_NSCR register if there is not any programming request anymore in the bank.
8. Optionally, lock the OTP block using LOCKBLi to prevent further data changes.

Note: Do not write twice an OTP 16-bit word, otherwise an ECC error is generated.

Writing OTP data at byte level is not supported, and generates a bus error.

To avoid data corruption, it is important to complete the OTP write process (for example by reading back the OTP value), before starting an option change.

Flash read-only area

The embedded flash memory offers a 2-Kbyte area to store read-only data. This area can be accessed through the AHB main port, and is protected by a robust ECC scheme, as detailed in *Section 7.3.8*.

The read-only information (programmed by STMicroelectronics) that can be used by the application software is detailed in *Table 47*.

Table 47. Read-only public data organization

Read-only data name	Address	Comment
Unique device ID	0x08FF F800	U_ID[31:0]
	0x08FF F804	U_ID[63:32]
	0x08FF F808	U_ID[96:64]
Flash memory size/ package	0x08FF F80C	Flash memory size[15:0] / Package code[15:0]
Reserved	0x08FF F810 to 0x08FF FFFF	Reserved information

7.3.10 Flash high-cycle data

The embedded flash memory offers up to 96 Kbytes (maximum) memory area with high-cycling capability (100 kcycles) to store data and emulate EEPROM. It can be accessed through the AHB system port from address 0x0900_0000 to 0x0901_7FFF (see *Figure 25*). It is mapped in the 8 (or 4) last sectors of Bank 1 and 2. This area is protected by a robust 6-bit ECC, enabling a 16-bit read and write granularity, at the expense of having sector size shrunk to 6 Kbytes.

A threshold per bank (EDATA(1/2)_STRT) is programmable to determine the beginning of the data flash area. By default, the whole memory is used for code.

For example, if 48 Kbytes of data are needed in Bank 1, set EDATA1_EN to 1, and EDATA1_STRT to 7. If no data are needed in Bank2, set EDATA2_EN to 0. In this case the data are accessible from address 0x0900_0000 to 0x0900_BFFF for Bank 1.

For greater efficiency, it is recommended to use sector on the other bank for flash high-cycle data, so that the application benefits from RWW capability of the dual-bank arrangement.

When SWAP_BANK feature is enabled, the banks are swapped: the flash high-cycle data in Bank 2 are accessible from 0x0900_000 to 0x0900_BFFF, and the data in Bank 1 are accessible from 0x0900_C000 to 0x0901_7FFF.

A bus error is generated on:

- Attempt to access an address between 0x0900_0000 to 0x0901_7FFF and this address is not valid (EDATA(1/2)_EN not enabled or EDATA(1/2)_STRT not correct).
- Attempt to fetch instructions from flash high-cycle data area.

Erasing the data area sector is possible by normal erase request for the corresponding user flash sector (120-127 for the STM32H562/563/573xx devices, 24-31 for the STM32H523/33xx devices).

Protections and security of high-cycle area are detailed in *Section 7.6.9*.

Figure 25. Flash high-cycle data memory map on 2-Mbyte devices

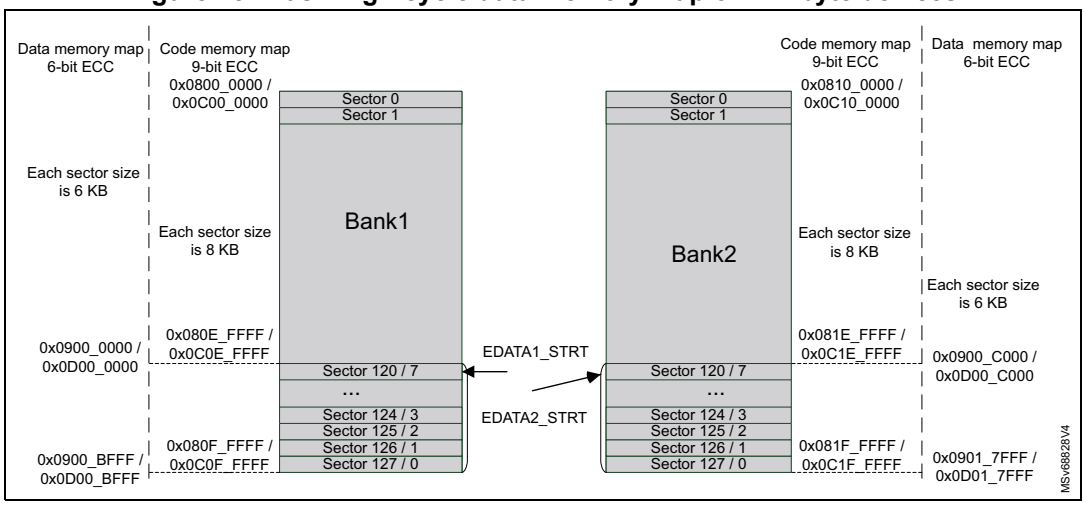


Figure 26. Flash high-cycle data memory map on 1-Mbyte devices

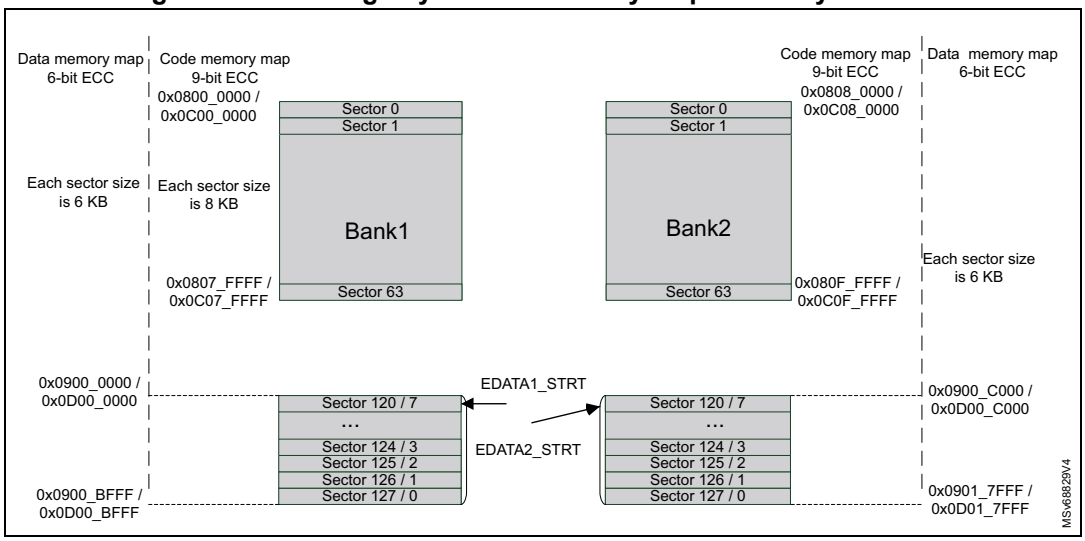


Figure 27. Flash high-cycle data memory map on 512-Kbyte devices

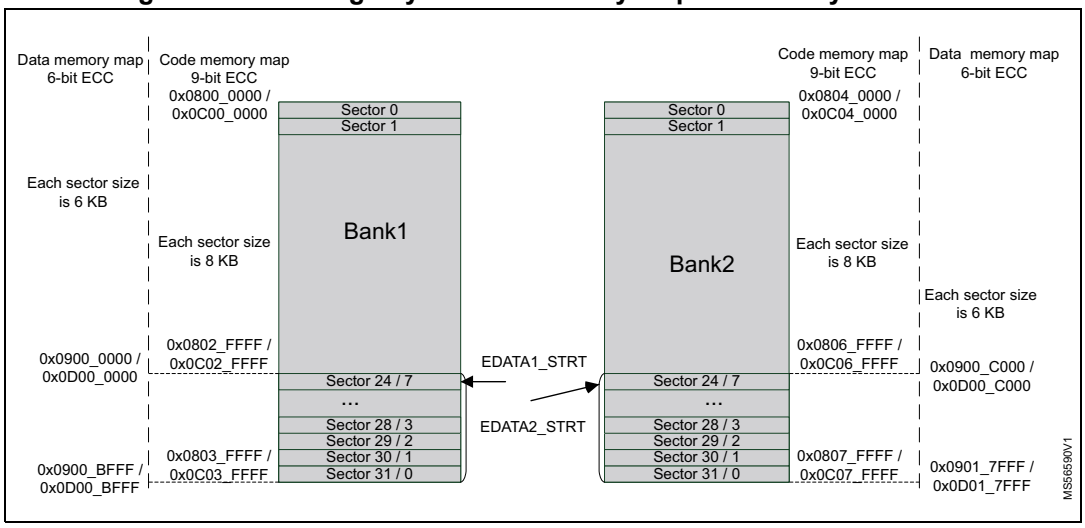
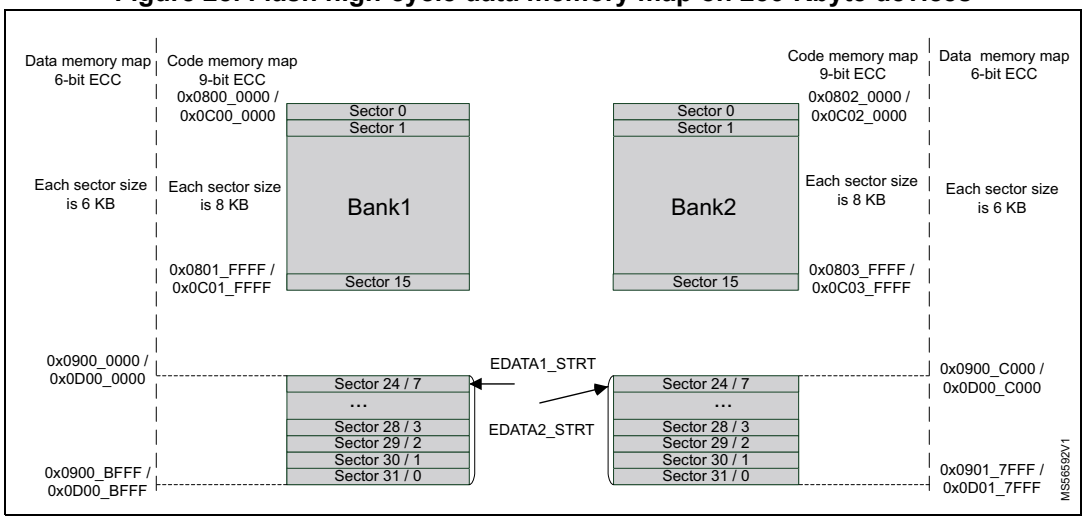


Figure 28. Flash high-cycle data memory map on 256-Kbyte devices



Note: When flash high-cycle data area on Bank1 is enabled, the code memory map is not continuous from Bank1 to Bank2 on 2M- and 512K-byte devices.

7.3.11 Flash bank swapping

Bank1 and Bank2 can be swapped for the user flash. This feature can be used after a firmware upgrade to restart the device on the new firmware. Bank swapping is an user option byte flag controlled by the SWAP_BANK bit of the FLASH_OPTCR register.

Bank specific settings for data area and security attributes follow the original bank and its contents. Control bit BKSEL always refers to physical bank, not the SWAP_BANK setting.

Table 48 shows the accessible memory, depending upon the SWAP_BANK bit value.

Table 48. Memory map and swapping options (STM32H562/563/573xx devices)

Area	Corresponding bank		Start address	End address	Size (bytes)	Region name
	SWAP_BANK = 0	SWAP_BANK = 1				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8 K	Sector 0
			0x0800 2000	0x0800 3FFF	8 K	Sector 1
			...	...	...	...
			0x080F E000	0x080F FFFF	8 K	Sector 127
	Bank2	Bank1	0x0810 0000	0x0810 1FFF	8 K	Sector 0
			0x0810 2000	0x0810 3FFF	8 K	Sector 1
			...	...	...	...
			0x081F E000	0x081F FFFF	8 K	Sector 127
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8 K	System 1 Sector 0
			0x0BF8 2000	0x0BF8 3FFF	8 K	System 1 Sector 1
			...	...	...	...
			0x0BF8 E000	0x0BF8 FFFF	8 K	System 1 Sector 7
	Bank2		0x0BF9 0000	0x0BF9 1FFF	8 K	System 2 Sector 0
			0x0BF9 0000	0x0BF9 3FFF	8 K	System 2 Sector 1
			...	...	...	...
			0x0BF9 E000	0x0BF9 FFFF	8 K	System 2 Sector 7

Table 49. Memory map and swapping options (STM32H523/533xx devices)

Area	Corresponding bank		Start address	End address	Size (bytes)	Region name
	SWAP_BANK = 0	SWAP_BANK = 1				
User main memory	Bank1	Bank2	0x0800 0000	0x0800 1FFF	8 K	Sector 0
			0x0800 2000	0x0800 3FFF	8 K	Sector 1
			⋮	⋮	⋮	⋮
			0x0803 E000	0x0803 FFFF	8 K	Sector 31
	Bank2	Bank1	0x0804 0000	0x0804 1FFF	8 K	Sector 0
			0x0804 2000	0x0804 3FFF	8 K	Sector 1
			⋮	⋮	⋮	⋮
			0x0807 E000	0x0807 FFFF	8 K	Sector 31

Table 49. Memory map and swapping options (STM32H523/533xx devices) (continued)

Area	Corresponding bank		Start address	End address	Size (bytes)	Region name
	SWAP_BANK = 0	SWAP_BANK = 1				
System memory	Bank1		0x0BF8 0000	0x0BF8 1FFF	8 K	System 1 Sector 0
			0x0BF8 2000	0x0BF8 3FFF	8 K	System 1 Sector 1
			...	...	...	...
			0x0BF8 E000	0x0BF8 FFFF	8 K	System 1 Sector 7
	Bank2		0x0BF9 0000	0x0BF9 1FFF	8 K	System 2 Sector 0
			0x0BF9 0000	0x0BF9 3FFF	8 K	System 2 Sector 1
			...	...	...	...
			0x0BF9 E000	0x0BF9 FFFF	8 K	System 2 Sector 7

The SWAP_BANK bit in FLASH_OPTCR register is loaded from the SWAP_BANK option bit only after system reset or POR.

To change the SWAP_BANK bit (for example to apply a new firmware update), follow the sequence below:

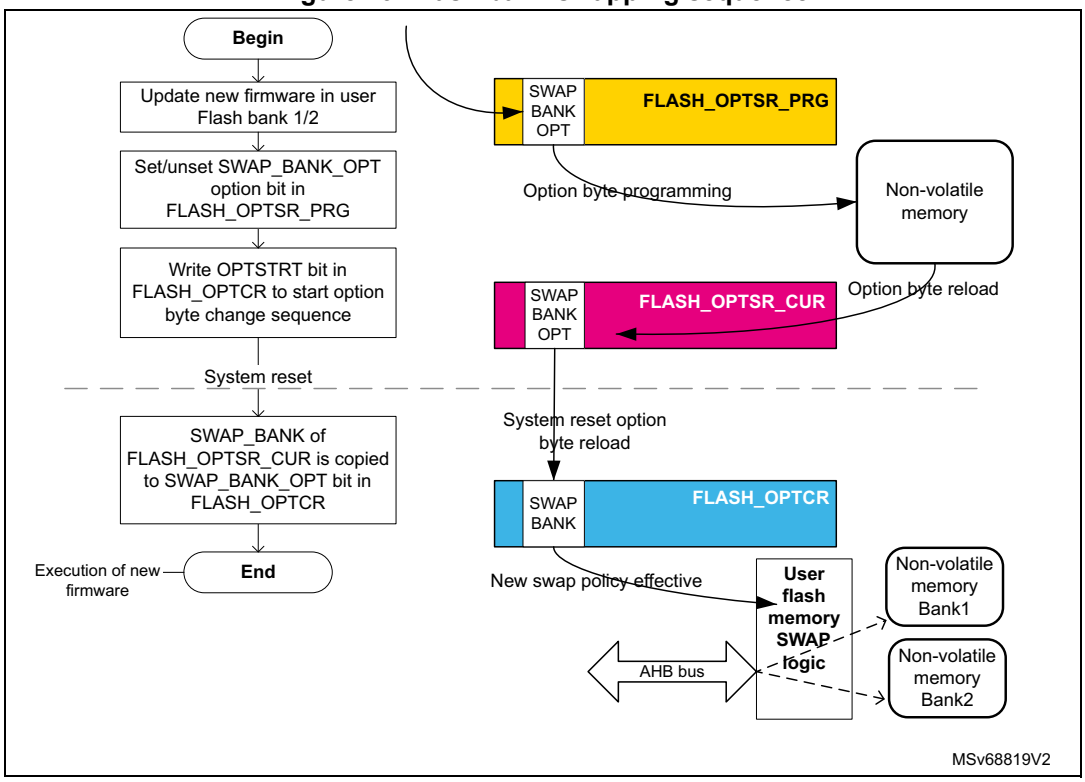
1. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_NS/SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NS/SECSR register.
2. Clear all error flags due to a previous operation.
3. Unlock OPTLOCK bit, if not already unlocked.
4. Set the new desired SWAP_BANK value in the FLASH_OPTSR_PRG register.
5. Start the option byte change sequence by setting the OPTSTRT bit in the FLASH_OPTCR register.
6. Once the option byte change has completed, FLASH_OPTSR_CUR contains the expected SWAP_BANK value, but SWAP_BANK bit in FLASH_OPTCR has not yet been modified and the bank swapping is not yet effective.
7. Force a system reset or a POR. When the reset rises up, the bank swapping is effective (SWAP_BANK value updated in FLASH_OPTCR) and the new firmware shall be executed.

Note: The SWAP_BANK bit in FLASH_OPTCR is read-only, and cannot be modified by the application software.

The SWAP_BANK option bit in FLASH_OPTSR_PRG can be modified whatever the product state. Instead of being locked by PRODUCT_STATE, it is locked by (NS/SEC)BOOT_LOCK User OB.

Figure 29 gives an overview of the bank swapping sequence.

Figure 29. Flash bank swapping sequence



7.3.12 FLASH reset and clocks

Reset management

The embedded flash memory can be reset by a core domain reset, driven by the reset and clock control (RCC). The main effects of this reset are the following:

- All registers, except for option byte registers, are cleared, including read and write latencies. If the bank swapping option is changed, it is applied.
- Most control registers are automatically protected against write operations. To unprotect them, new unlock sequences must be used as described in *Section 7.6.7*.

The memory can be reset by a power-on core domain reset, driven by the reset and clock control (RCC). When the reset falls, all option byte registers are reset. When the reset rises up, the option bytes are loaded, potentially applying new features. During this loading sequence, the device remains under reset, and the memory is not accessible.

The reset signal can have a critical impact on the memory: the content is not guaranteed if a device reset occurs during a write or erase operation.

Reset occurring during flash operation

If a reset occurs during a flash operation (programing, erase or option change), the content of the memory is not guaranteed. It is mandatory for integrity to restart the operation. The status register FLASH_OPSR gives information about operations interrupted by a reset.

FLASH_OPSR.CODE_OP gives opcode of operation. *Table 50* indicates how to use FLASH_OPSR, and which operation is required.

Table 50. Recommended reactions to FLASH_OPSR contents

CODE_OP	Operation interrupted	OTP_OP	SYSF_OP	BK_OP	DATA_OP	Flash area	ADDR_OP min	ADDR_OP max	Recommended action
0x000	No operation ongoing while reset	0	0	0	0	-(1)	-	-	No extra action
0x001	Write operation	0	0	0/1	0	User flash	0x0000	0xFFFF	Erase sector and rewrite
		0	1	0/1	0	System flash	0x0000	0x0FFF	
		1	0	0	0	OTP	0x0600	0x07FF	
		1	0	1	0	OBKeys(2)	0x0000	0x03FF	
		0	0	0/1	1	Data area(3)	0xF000	0xFFFF	
0x010	OBK alternate sector erase	0	0	1	0	OBKeys	0	0	Relaunch the alternate sector erase
0x011	Sector erase(4)	0	0	0/1	0	User flash	0x0000	0xFFFF	Relaunch sector erase
		0	1	0/1	0	System flash	0x0000	0x0FFF	
0x100	Bank erase	0	0	0/1	0	User flash	-	-	Relaunch bank erase
0x101	Mass erase	0	0	0	0		-	-	Relaunch mass erase
0x110	Option change	0	0	0	0	User configuration	-	-	New attempt on option change
0x111	OBK swap sector	0	0	1	0	OBKeys	-	-	Erase alternate sector, reprogram new OBKeys and relaunch swap

1. Dash represents “does not matter”.
2. Depends on current OBK sector.
3. Depends on EDATA setting in the OB.
4. Addresses indicated are aligned to sector start, data area sectors are erased by erase request to corresponding user flash memory sector.

Clock management

The memory uses the microcontroller system clock (sys_ck), here the AHB interface clock.

7.4 FLASH option bytes

7.4.1 About option bytes

The memory includes a set of nonvolatile option bytes. They are loaded at power-on reset and can be read and modified only through configuration registers.This section details:

- when option bytes are loaded
- how application software can modify them
- their detailed list, together with their initial values (before the first option byte change, user default configuration).

7.4.2 Option bytes loading

There are multiple ways of loading the option bytes:

- **Power-on wake-up**

  When the device is first powered, the embedded flash memory automatically loads all the option bytes. During the option byte loading sequence, the device remains under reset and the embedded flash memory cannot be accessed.
- **Wake-up from system Standby**

  When the core power domain, which contains the embedded flash memory, is switched from Standby mode to Run mode, the embedded flash memory behaves as during a power-on sequence. **During loading time the device is not under reset, unlike power-on sequence.**
- **Dedicated option byte reloading by the application**

  When the user application successfully modifies the option byte content through the memory registers, the nonvolatile option bytes are programmed and the memory automatically reloads all option bytes to update the option registers.

Note: The option byte read sequence is protected by error correction code.In case of error, the option bytes are loaded with default values (see Section 7.4.3), different from the initial values (user default configuration), and more restrictive.

7.4.3 Option bytes modification

Changing user option bytes

A user option byte change operation can be used to modify the configuration and the protection settings saved in the nonvolatile option byte area.

There are several rules enforced when attempting to change a user option byte, they are summarized in *Section 7.4.7*. Failing to stick to those rules usually results in errors, described in *Section 7.9.12*.

The embedded flash memory features two sets of option byte registers:

- The first register set contains the current values of the option bytes. Their names have the _CUR extension. All these registers are read-only. Their values are automatically loaded from the nonvolatile memory after power-on reset, wake-up from system standby or after an option byte change operation.
- The second register set allows the modification of the option bytes. Their names contain the _PRG extension. All “_PRG” registers can be accessed in read/write mode.

When the OPTLOCK bit in FLASH_OPTCR register is set, it is not possible to modify the FLASH_XXX_PRG registers.

When OPTSTRT bit is set to 1, the memory checks the programming sequence (PGSERR) and the conditions described in *Section 7.4.7* (OPTCHANGEERR). If no error has been detected (PGSERR/OPTCHANGEERR), the flash interface launches the option byte modification, and updates the option byte registers with _CUR extension.

If one of the condition described in *Section 7.4.7*, *Section 7.9.12* or *Section 7.9.5*, alternatively *Section 7.9.4* is not respected, the memory aborts the option byte change operation. In this case, the FLASH_XXX_PRG registers are not overwritten by the current option value. The user application can check what was wrong in their configuration.

Unlocking the option byte modification

After reset, the OPTLOCK bit is set to 1 and the FLASH_OPTCR is locked. As a result, the application software must unlock the option configuration register before attempting to change the option bytes. The FLASH_OPTCR unlock sequence is described in *Section 7.6.7*.

Option bytes modification sequence

To modify user option bytes, follow the sequence below:

1. Check that no memory operations are ongoing by checking the BSY bit in the FLASH_NS/SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_NS/SECSR register.
2. Check the data buffer is empty (DBNE = 0) in FLASH_NS/SECSR register.
3. Clear all error flags due to a previous operation.
4. Unlock FLASH_OPTCR register as described in *Section 7.6.7*, unless the register is already unlocked.
5. Write the desired new option byte values in the corresponding option registers (FLASH_XXX_PRG).
6. Set the option byte start change OPTSTRT bit to 1 in the FLASH_OPTCR register.
7. Wait until BSY bit is cleared in FLASH_NS/SECSR register.
8. OPTSTRT bit is cleared automatically at the end of the sequence (or in case of error).
9. Reset the device. This step is always recommended, becomes mandatory when one of the following option bytes is impacted:
   a) SECBOOTADDR
   b) NSBOOTADDR
   c) TZEN
   d) BOOT_UBE

Note: If a reset or a power-down occurs while the option byte modification is ongoing, the original option byte value is kept. A new option byte modification sequence is required to program it.

Option bytes overview

Table 51 lists all the user option bytes managed through the memory registers, as well as the initial values before the first option byte change (user default configuration).

Table 51. Option bytes organization

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_ OPTSR	SWAP_BANK	Res.	BOOT_UBE								IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV	PRODUCT_STATE								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV	
	0	0	1	0	1	1	0	1	0	0	1	1	0	0	0	0	1	1	1	0	1	1	0	1	1	1	0	1	1	0	0	0
FLASH_ OPTSR2	TZEN								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPD_DIS	Res.	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	SRAM2_RST	SRAM13_RST	Res.	Res.
	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0	0
FLASH_ BOOTR	SECBOOTADD[23:8]																SECBOOTADD[7:0]								SECBOOT_LOCK							
	0x0C00																0x00								0xC3							
FLASH_ NSBOOTR	NSBOOTADD[23:8]																NSBOOTADD[7:0]								NSBOOT_LOCK							
	0x0800																0x00								0xC3							
FLASH_ SECWM1R	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END							Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
FLASH_ SECWM2R	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT						
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
FLASH_ WRPSGN1R	WRPSG[127:124]	WRPSG[123:120]	...												WRPSG[71:68]	WRPSG[67:64]	WRPSG[63:60]	WRPSG[59:56]	...												WRPSG[7:4]	WRPSG[3:0]	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
FLASH_ WRPSGN2R	WRPSG[127:124]	WRPSG[123:120]	...												WRPSG[71:68]	WRPSG[67:64]	WRPSG[63:60]	WRPSG[59:56]	...												WRPSG[7:4]	WRPSG[3:0]	
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 51. Option bytes organization (continued)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_ OTPBLR	LOCKBL																															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_ EDATA1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA_EN1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECTOR_START_1		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_ EDATA2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA_EN1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECTOR_START_2		
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FLASH_ NSEPOCHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NS_EPOCH																							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_ SECEPOCHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_EPOCH																							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_ HDP1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END								HDP1_STRT							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FLASH_ HDP2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END								HDP2_STRT							
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

7.4.4 Description of user and system option bytes

The general-purpose option bytes that can be used by the application are listed below:

- Watchdog management
  - IWDG_STOP: independent watchdog IWDG (also known as WDGLS_CD) counter active in Stop mode if 1 (stop counting or freeze if 0)
  - IWDG_STDBY: independent watchdog IWDG (also known as WDGLS_CD) counter active in Standby mode if 1 (stop counting or freeze if 0)
  - IWDG_SW: hardware (0) or software (1) IWDG (also known as WDGLS_CD) watchdog control selection

Note: If the hardware watchdog "control selection" feature is enabled (set to 0), the watchdog is automatically enabled at power-on, thus generating a reset unless the watchdog key register is written to or the down-counter is reloaded before the end-of-count is reached.

Depending on the configuration of IWDG_STOP and IWDG_STDBY options, the IWDG can continue counting (1) or not (0) when the device is, respectively, in Stop or Standby mode.

When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

- Reset management
  - BOR_LEV: Brownout level option, indicating the supply level threshold that activates/releases the reset
  - BORH_EN: enabling a high BOR level
  - NRST_STDBY: generates a reset when entering Standby mode if cleared to 0
  - NRST_STOP: generates a reset when entering Stop mode if cleared to 0.

Note: Whenever a Standby (Stop) mode entry sequence is successfully executed, the device is reset instead of entering Standby (Stop) mode if NRST_STDBY (NRST_STOP) is cleared to 0.

- Device options
  - IO_VDDIO2_HSLV: enables the configuration of pads below 2.7 V for VDDIO2 power rail if set to 1
  - IO_VDD_HSLV: enables the configuration of pads below 2.7 V for VDD power rail if set to 1
  - USBPD_DIS: bit to disable USB PD

At delivery, the values programmed in the user option bytes are the following:

- Watchdog management
  - IWDG (also known as WDGLS_CD) active in Standby and Stop modes: 0x1
  - IWDG (also known as WDGLS_CD) not automatically enabled at power-on: 0x1
- Reset management
  - BOR: brownout level option (reset level = 2.1 V): 0x0. A reset is not generated when the device enters Standby, Stop low-power mode (value = 0x1)
- Device working in the full voltage range with I/O speed optimization at low-voltage disabled (IO_VDDIO2_HSLV = 0 and IO_VDD_HSLV = 0)

Refer to *Section 7.11* for details.

7.4.5 Description of data protection option bytes

The option bytes that can be used to enhance data protection are listed below:

- PRODUCT_STATE[7:0]: A product life cycle state (see *Section 7.6.11* for details).
- WRPSGn1/2: write protection option of the corresponding group of four consecutive sectors in Bank1 (respectively Bank2). It is active low. Refer to *Section 7.6.8* for details.
  - Bit N: Group embedding sectors 4 x N to 4 x N + 3
- SECWMx: TrustZone® secure only watermark area definition (refer to *Section 7.6.1* for details).
  - SECWM1_STRT (respectively SECWM1_END) contains the first (respectively last) sector of the secure access only zone in Bank1
  - SECWM2_STRT (respectively SECWM2_END) contains the first (respectively last) sector of the secure access only zone in Bank2.
- TZEN: this nonvolatile option can be used by the application to activate the secure access mode, as described in *Section 7.6*. For security reasons the TZEN is stored with redundancy on 8 bits. While the TZEN OB value is set immediately after

programming, to actually use TZ features, reset is required to set the TZ_STATE in SBS.

When TZEN is activated, secure watermark settings and secure boot address can be reset to default to prevent a deadlock in case the previous values were not correct.

- HDPx: Secure hide protection - HDPL exclusive area control in the user flash memory.

When factory programmed values of the data protection option bytes are the following:

- Product state depends on sales type
- Flash bank erase operations do not impact watermarked secure data areas
- Secure watermark areas protections disabled (start addresses higher than end addresses)
- Write protection disabled (all option byte bits set to 1)
- TrustZone Secure access mode disabled (TZEN option byte value = 0xC3)

Refer to *Section 7.11* for details.

7.4.6 Description of boot address option bytes

Below the list of option bytes that can be used to configure the appropriate boot address for an application:

- PRODUCT_STATE
- BOOT_UBE: Selects either ST-iRoT or User flash as default boot address. Also used by RSS(SFI) to choose either Bootloader or ST-iRoT as next stage. Available only on products embedding cryptographic acceleration (STM32H533/573xx).
- NSBOOTADD: Selects default boot address when TZ is disabled.
- SECBOOTADD: Selects default boot address when TZ is enabled.
- (NS/SEC)BOOT_LOCK: Protects the boot configuration from further modification attempts.
- SWAP_BANK: bank swapping option, set to 1 to swap user flash banks after boot (see *Section 7.3.11*). If BOOT_LOCK corresponding to TZEN state (SEC/NS) is active, the value in SWAP_BANK is fixed, read-only.

When STMicroelectronics delivers the device, the PRODUCT_STATE is Open, BOOT_LOCK is not set (0xC3) and the BOOT_UBE is set to user flash (0xB4). Addresses are SECBOOTADD = 0x0C00 0000, NSBOOTADD = 0x0800 0000.

Refer to *Section 7.11* for details.

7.4.7 Specific rules for modifying option bytes

On top of OPTLOCK bit and register access rules, there are other protections for selected security-sensitive option byte fields.

Different option bytes can be modified simultaneously, but if they rely on each other for protection, both states are checked. For example, when trying to modify PRODUCT_STATE and TZEN simultaneously, resulting state must be coherent with the rules. With few exceptions, listed in this section, failing to uphold the rules results in raising OPTCHANGEERR flag (*Section 7.9.12* for additional details).

Table 52. Specific modifying rules

Option byte	HDPL	TZ secure	Value	Product state
PRODUCT_STATE	1(1)	Some values only when TZ is enabled	Set of possible transitions	Set of possible transitions (*Table 45*)
HDP	1(2)	(3)	-	Refer to *Table 53*
TZEN	-	-	-	Regression, Open or Provisioning
EPOCH	-	-	PRG > CUR	Regression
EPOCH_NS	-	-	PRG > CUR	NS-Regression, Regression
SECWM	-	Secure mode only(2)	-	Refer to *Table 53*
BOOT_UBE	-	-	-	Open or Provisioning, SECBOOT_LOCK disabled
SECBOOTADD	-	Secure mode only(2)	-	
NSBOOTADD	-	-	-	Open or Provisioning, NSBOOT_LOCK disabled
LOCKBL	-	-	One way switch(2)	-
SWAP_BANK	-	Fixed if BOOT_LOCK corresponding to TZEN is set	-	-
SECBOOT_LOCK	-	Secure mode only(2)	-	Open, Regression or Provisioning to unlock
NSBOOT_LOCK	-	-	-	

1. Most transitions are possible regardless of HDPL, only a few require specific HDPL, see text below.
2. No OPTCHANGEERR raised in violation of this.
3. Dash means there is no limitation from this side.

Even in the closed PRODUCT_STATE progression, some OBs can still be modified, if all the other constraints are satisfied. An overview is presented in *Table 53*.

Table 53. OB modifiable in closed product

PRODUCT_STATE	OBs that can be modified
TZ-Closed	SWAP_BANK, LOCKBL, PRODUCT_STATE, SECWM, HDP
Closed	SWAP_BANK, LOCKBL, PRODUCT_STATE
Locked	SWAP_BANK, LOCKBL

Specific rules must be respected to update the following OB:

- **PRODUCT_STATE**

  PRODUCT_STATE transitions follow a state machine with two types of transitions. Locking down the product is allowed without restriction. Opening the product is possible only using a debug interface and following a digital signature verification. The regression to a less secure state results in the erase of the protected content. More details are given in *Section 7.6.11*.

  Summary: Selected changes are possible only in HDPL1. Some states are accessible only when TZEN is enabled.

- **HDP**

  Can be modified only in HDPL1. Do not set it to overlap with flash high-cycle data area.
- **TrustZone access mode (TZEN)**

  Can be changed only in Open, Regression and Provisioning.
- **SEC_EPOCH and NS_EPOCH**

  The value programmed must be greater than current value.The increment is done in specific product states: Regression and NS-Regression.
- **Secure watermark area (SECWM1/2_STRT and SECWM1/2_END)**

  Can be changed only in secure mode (TZ_state = 0xB4). Automatically reset to default value when TZEN is enabled. Do not set it to to overlap with flash high-cycle data area.
- **BOOT_UBE**

  Available only on cryptography enabled devices. Can be changed only in product states open for debug like Open and also in Provisioning. SECBOOT_LOCK must be disabled to change.
- **SECBOOTADD**

  Can be changed only in Open and Provisioning. Locked by SECBOOT_LOCK. Automatically reset to default value when TZEN is enabled.
- **NSBOOTADD**

  Can be changed only in Open and Provisioning. Locked by NSBOOT_LOCK.
- **LOCKBL**

  Can be changed freely only in one direction. A permanent irreversible switch.
- **SWAP_BANK**

  Not modifiable when both TZEN and SECBOOT_LOCK are 0xB4 (set) or when TZEN = 0xC3 (disabled) and NSBOOT_LOCK is active (0xB4).
- **SECBOOT_LOCK**

  Can be changed only freely in locking direction. Unlock is possible in Open, Provisioning and Regression.
- **NSBOOT_LOCK**

  Can be changed only freely in locking direction. Unlock is possible in Open, Provisioning and Regression.

Note: For all user option bytes above: default values are loaded and Tamper is signaled when a double ECC error occurs during OBL.

7.5 Option bytes key (OBK) management

This section describes the key storage tied with SAES and OBK-HDPL value set by the SBS (refer to *Section 14.3.7: SBS hardware secure storage control*). The OBKs are stored in 128 bits and protected by 9-bit ECC (SEC/DED). They are written alternatively in two sectors.

7.5.1 OBK loading

OBKs are memory mapped, and accessible through the main AHB bus (C-Bus), starting from address 0x0FFD 0000. A 9-bit ECC is associated to each 128-bit data flash word.

Note: Read access is allowed in the current and alternate sector, except for the last address of the current and alternate sector. In case of read access, no error is reported but the read data is always 0x0 (see Section 7.6.12).

7.5.2 OBK access per HDPL level

Table 54 describes the option byte key areas. The key storage is not dedicated to a particular key, and the usage is defined by the application.

An option byte key can be accessed only if the OBK-HDPL (set in SBS) matches the HDPL associated to the storage offset, indicated below (refer to *Section 14.3.7: SBS hardware secure storage control*). If this is not the case, an OBKERR error is raised.

Table 54. Option bytes key area

OBK-HDPL	Address offset start	Address offset end	Comment
-	0x0000	0x00FF	Reserved
1	0x0100	0x08FF	OEM iRoT keys
2	0x0900	0x0BFF	uRoT, OS or secure application
3	0x0C00	0x17FF	HDPL 3 secure keys
3	0x1800	0x1FEF	HDPL 3 non-secure keys

7.5.3 OBK programming sequence

The sequence to write OBK is the same as a write to main flash except that the “alternate sector” (ALT_SECT) bit must be set to write in the alternate OBK sector. The ALT_SECT bit is checked each time a write access is received. If it does not match the one stored in the write buffer, an error is raised.

Note: Before writing in the alternate OBK sector the user must ensure that the whole key space (128, 256, or 512 bits) is free, by reading the alternate OBK sector.

If the alternate sector is not free, erase the sector using the command ALT_SECT_ERASE.

Secure programming method

Using the secure register if TrustZone is active (TZ_STATE = 0xB4):

1. Program the ALT_SECT bit.
2. Make sure protection mechanism does not prevent programming (TZ, PRIV and OBK-HDPL).
3. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_SECSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SECSR register.
4. Check and clear all the secure error flags due to previous operation.
5. Unlock the FLASH_SECCR register only if this register is not already unlocked, as described in *Section 7.6.7*.
6. Enable write operations by setting the SECPG bit in the FLASH_SECCR register.
7. Write one flash-word corresponding to 16-byte data starting at 16-byte aligned address.

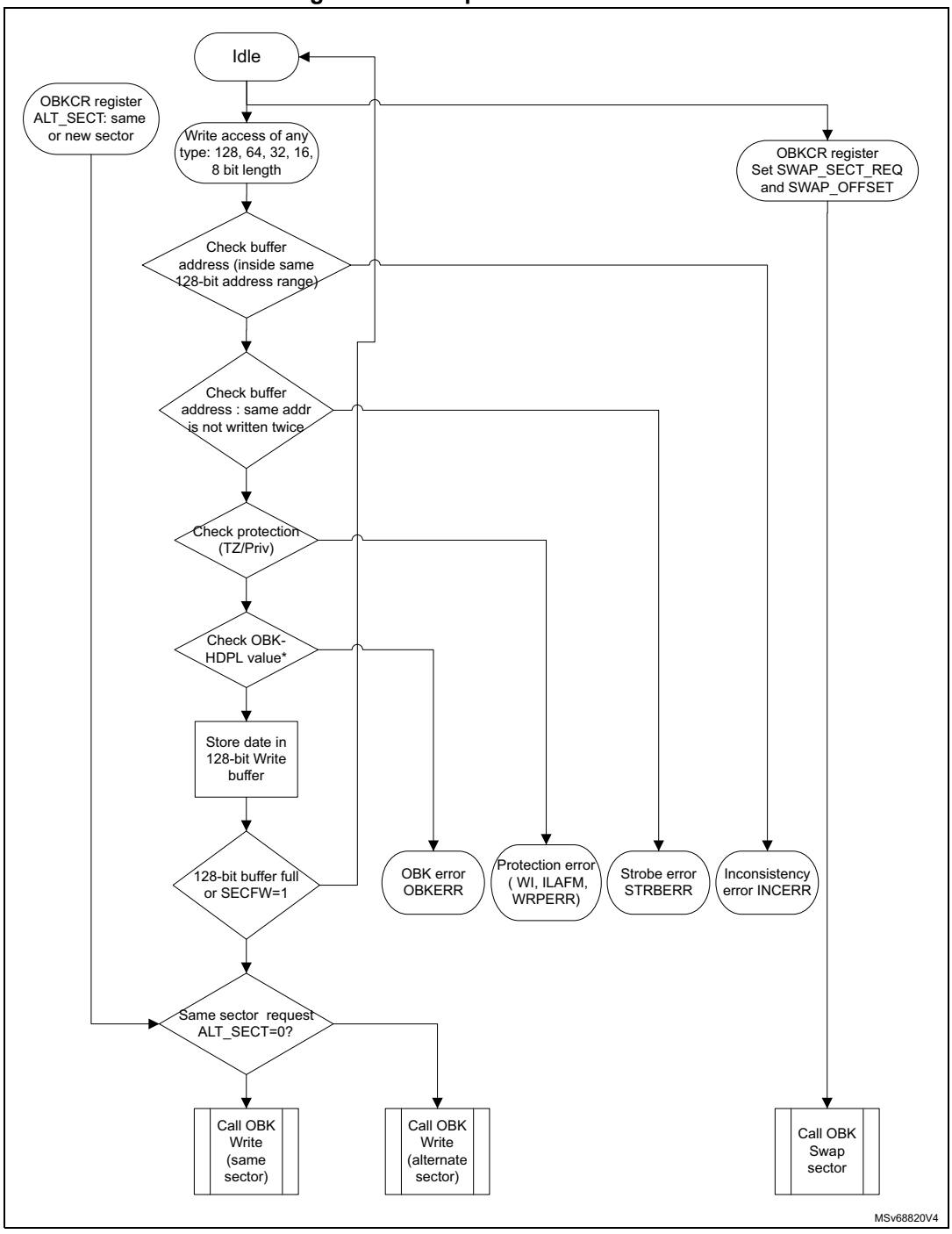
Note: WBNE flag indicates if the 128-bit write buffer is waiting for new data.

Note: No erase request, options change request, OBK swap sector or OBK alternate sector erase is allowed between first write and the completion of flash write operation.

8. Wait for the BSY bit to be cleared in the corresponding FLASH_SECSR register.
9. Clear PG bit in FLASH_SECCR register if there is not any programming request anymore in the bank.

If step 7 is executed incrementally (byte per byte), the write buffer can become partially filled. In this case the application software can decide to force-write what is stored in the write buffer by using FW bit in FLASH_SECCR register. In this particular case, the unwritten bits are automatically set to 1. If no bit in the write buffer is cleared to 0, the FW bit has no effect.

Figure 30. OBK protection checks

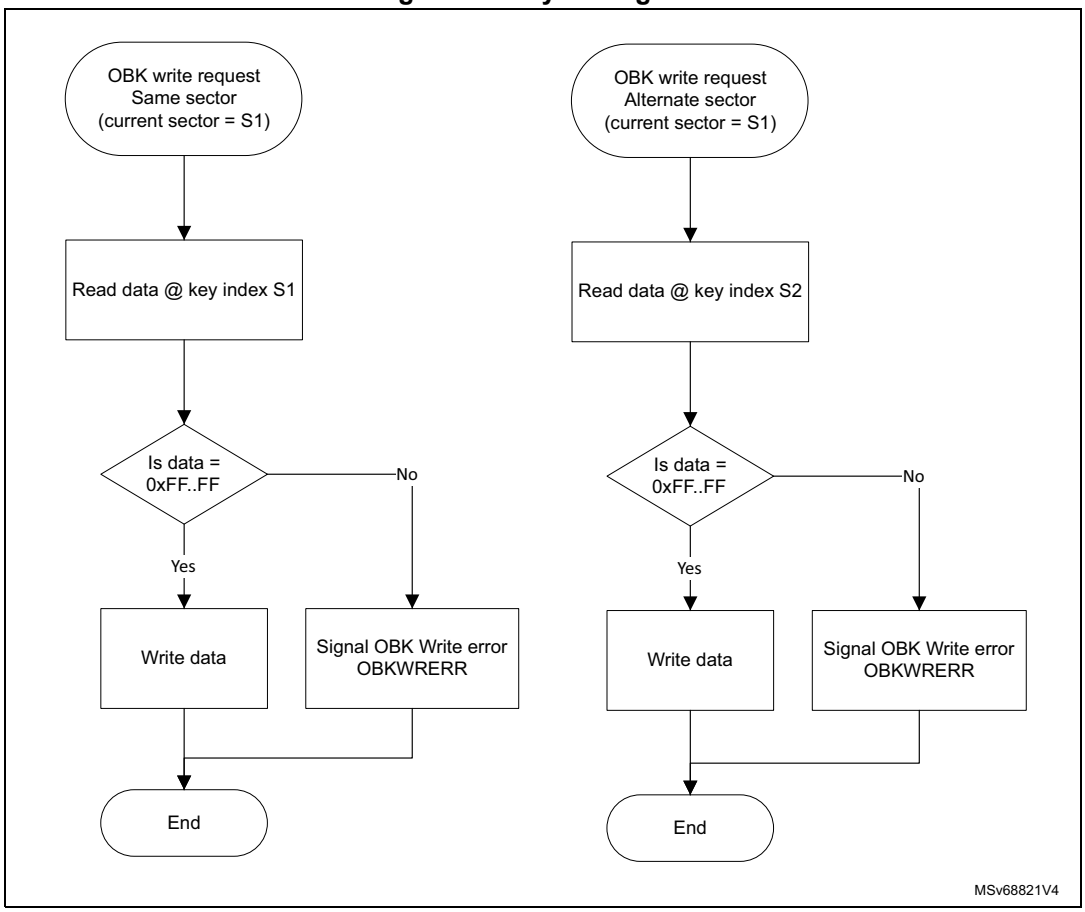


7.5.4 OBK programming finite state machine

When the OBK write buffer is filled, the OBK programing is launched automatically in the current OBK sector or in the alternate OBK sector, depending upon the ALT_SECT bit.

If the address is not virgin, the OBKWRERR is raised.

Figure 31. Key writing flow



7.5.5 OBK swap sector

The user can request a swap of the option byte keys by setting the bit SWAP_SECT_REQ in the NS/SECOBKCFGR register. The number of keys swapped is defined by the SWAP_OFFSET in NS/SECOBKCFGR register.

Swap is limited by OBK-HDPL value, the following SWAP_OFFSET values are permitted:

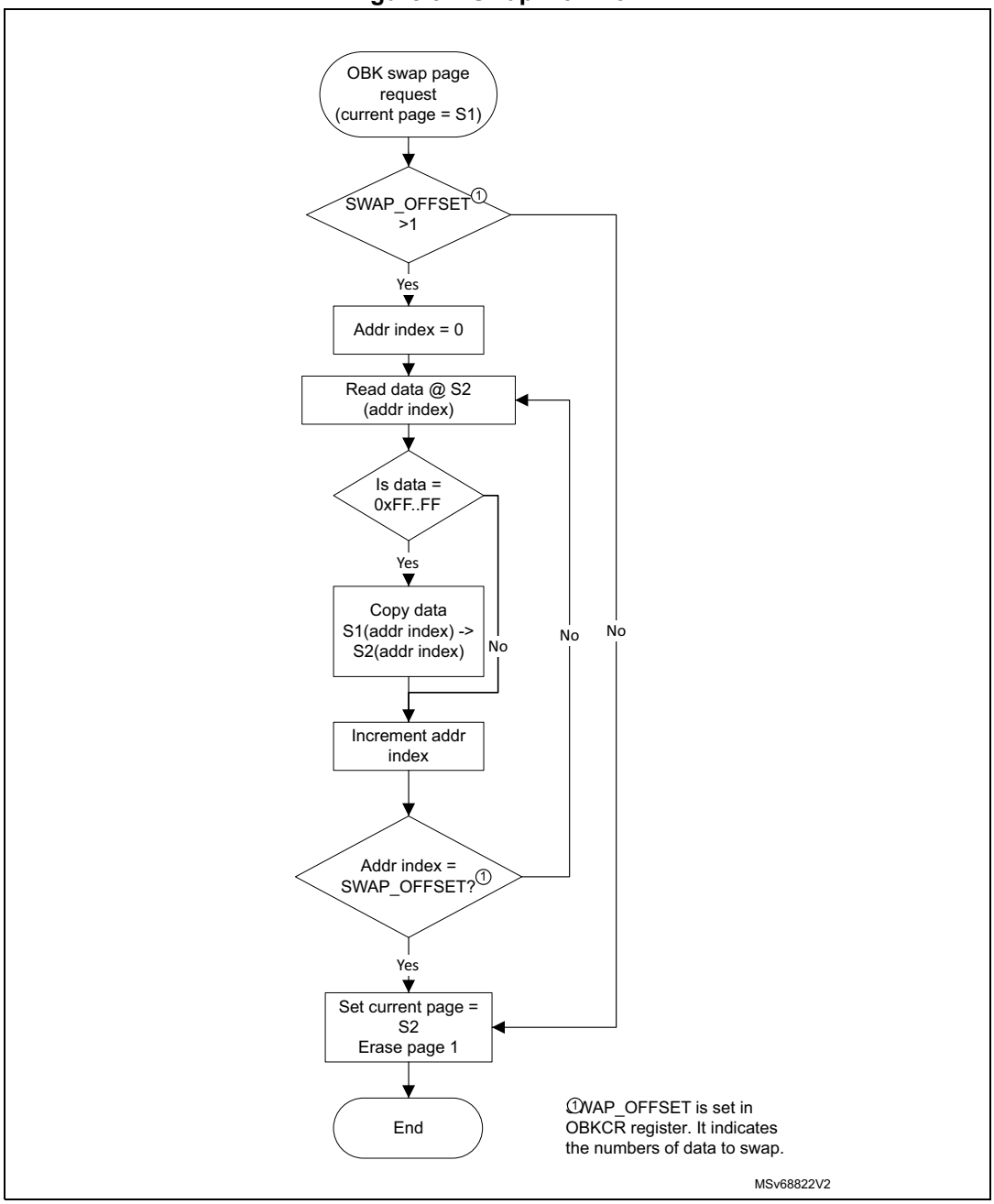
- OBK_HDPL1_OFFSET = 144
- OBK_HDPL2_OFFSET = 192
- OBK_HDPL3SEC_OFFSET = 384
- OBK_HDPL3NS_OFFSET = 511

The programming sequence is the following:

1. Check that no memory operations are ongoing by checking the BSY and DBNE bit in the FLASH_SEC/NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SEC/NSSR register.
2. Check and clear all the secure/non-secure error flags due to previous operation.
3. Unlock NS/SECOBKCFGR register if not unlocked already thanks to FLASH_NS/SECOBKKEYR.

4. Define the numbers of keys to be swapped thanks to SWAP_OFFSET field in NS/SECOBKCFGR. SWAP_OFFSET must be equal to or greater than OBK_HDPL<N-1>_OFFSET.
5. Set bit SWAP_SECT_REQ in NS/SECOBKCFGR.
6. Wait for BSY bit to be cleared. In case of error OBKERR flag is set.
7. SWAP_SECT_REQ is cleared automatically at the end of the operation or in case of error.

Figure 32. Swap workflow



While the OBK SWAP operation is ongoing, the BSY flag is set and no other operation (write, erase, user opt change) can be launched in parallel.

7.5.6 OBK alternate sector erase

The user can request the erase of the alternate OBK sector by setting the bit ALT_SECT_ERASE in the FLASH_NS/SECOBKCFGR register.

The programming sequence must be:

1. Check that no memory operations are ongoing by checking the BSY and DBNE bits in the FLASH_SEC/NSSR register and that the write buffer is empty by checking the WBNE bit in the FLASH_SEC/NSSR register.
2. Check and clear all the secure/non-secure error flags due to previous operation.
3. Set bit ALT_SECT_ERASE in NS/SECOBKCFGR.
4. Wait for BSY bit to be cleared.
5. ALT_SECT_ERASE is cleared automatically at the end of the operation or in case of error.

7.6 FLASH security and protections

To protect sensitive information against unwanted operations (such as reading confidential areas, illegal programming of immutable sectors, or malicious flash memory erasing), the FLASH implements the following protection mechanisms:

- TrustZone backed watermark and block security protection
- Temporal isolation protection (HDP)
- Configuration protection
- User flash write protection
- Device nonvolatile security life cycle and application boot state management
- OTP locking

The microcontroller can have TrustZone active (TZ_STATE = 0xB4) or not (TZ_STATE = 0xC3).

While TrustZone is active the flash memory can be accessed in four basic modes:

- non-secure and unprivileged
- non-secure and privileged
- secure and privileged
- secure and unprivileged

While TrustZone is disabled, only two modes are possible:

- non-secure and unprivileged
- non-secure and privileged

The flash interface evaluates the access restrictions in the following order:

1. TrustZone security
2. Write protection (for write access)
3. HDP
4. Privilege

7.6.1 TrustZone security protection

The global TrustZone system security is activated by setting the TZEN option in the FLASH_OPTSR2_PRG register. The actual state of TZ is however determined by the SBS and the setting of TZ_STATE (refer to *Section 14.3.5: SBS boot control*).

When TrustZone is active (TZ_STATE = 0xB4), additional security features are available:

- Secure watermark-based user options bytes defining secure areas
- Secure or non-secure block-based areas can be configured on-the-fly after reset. This is a volatile secure area
- Additional product states associated with **closed_secure** provide protection of secure domain against external access (debug or bootloader)
- Erase or program operation can be performed in secure or non-secure mode with associated configuration bit. When the TrustZone is disabled (TZ_STATE = 0xC3), the above features are deactivated and all secure registers are RAZ/WI.

Activating TrustZone security

When the TrustZone is activated (TZ_STATE = 0xB4), the SECBOOTADD and the secure watermark areas are set to a secure default value. Default settings are also used in case of corrupted configuration. In this default state, user flash memory is secure (see *Table 55*).

Table 55. Default secure watermark

Secure watermark in case of configuration error	Security attribute
SECWMx_STRT = 0 SECWMx_END = 0x7F(1)	All flash memory is secure

1. Default and maximum values depend upon the total number of sectors in the user flash memory.

Illegal access generation

A non-secure access to a secure flash memory area is RAZ/WI, it generates an illegal access event. An illegal access interrupt is generated if the FLASHIE illegal access interrupt is enabled in the TZIC_IER2 register.

A non-secure access to a secure flash register generates an illegal access event. An illegal access interrupt is generated if the FLASH_REGIE illegal access interrupt is enabled in the TZIC_IER2 register.

Table 56. Flash memory TZ protection summary

TZ protection (TZ_STATE = 0xB4)		Main flash	
		Non-secure sector	Secure sector
Secure	Fetch	BUS ERROR	OK
	Read	RAZ, ILAFM	
	Write	WI, WRPERR, ILAFM	If not WRP: OK else: WI, WRPERR
	Erase		

Table 56. Flash memory TZ protection summary (continued)

TZ protection (TZ_STATE = 0xB4)		Main flash	
		Non-secure sector	Secure sector
Non-secure	Fetch	OK	BUS ERROR
	Read		RAZ, ILAFM
	Write	If not WRP: OK else: WI, WRPERR	WI, WRPERR, ILAFM
	Erase		

Table 57. TZ protection and bank or mass erase summary

TZ protection (TZ_STATE = 0xB4)	Main flash		
	Non-secure flash	Secure flash	Mix of secure and non-secure
MES	WI, WRPERR, ILAFM	OK	WI, WRPERR, ILAFM
MENS	OK	WI, WRPERR, ILAFM	WI, WRPERR, ILAFM

Deactivate TrustZone security

Reversing of the TZ setting in OB (TZEN) is possible when the product state is ST-RoT-Ready or Provisioning. Deactivating TZEN from other states requires full regression (Regression state).

When the TrustZone is deactivated (TZ_STATE = 0xC3) after option bytes loading, the following security features are deactivated:

- Watermark-based secure area (refer to *Watermark-based secure flash area protection*)
- block-based secure area (refer to *Section 7.6.3*)
- Secure interrupts (refer to *Section 7.9.11*)
- All secure registers are RAZ/WI.

Watermark-based secure flash area protection

When TrustZone security is active (TZ_STATE = 0xB4), a part of the flash memory can be protected against non-secure read and write accesses. Up to two different nonvolatile secure areas can be defined by option bytes and can be read or written by a secure access only: one area per bank can be selected with a sector granularity.

The secure areas are defined by a start sector offset and end sector offset using the SECWMx_STRT and SECWMx_END option bytes. These offsets are defined in the secure watermark registers address registers *FLASH security watermark for Bank1 (FLASH_SECWM1R_CUR)*, *FLASH security watermark for Bank2 (FLASH_SECWM2R_CUR)* and modified by corresponding FLASH_XXX_PRG registers.

The SECWMx_STRT and SECWMx_END option bytes can be modified only by secure firmware.

Table 58. Secure watermark-based area

Secure watermark option bytes values (x = 1, 2)	Secure watermark protection area
SECWMx_STRT > SECWMx_END	Nonsecure area
SECWMx_STRT = SECWMx_END	One sector defined by SECWMx_STRT is secure watermark-based protected
SECWMx_STRT < SECWMx_END	The area between SECWMx_STRT and SECWMx_END is secure watermark-based protected

Caution: Switching a memory area from secure to nonsecure does not erase its content. The user secure software must perform the needed operation to erase the secure area before switching an area to non-secure attribute whenever is needed. It is also recommended to flush the instruction cache.

7.6.2 Hide protection (HDP)

The HDP area is independent of the flash watermark-based secure area. One nonvolatile secure hide protection (HDP) area per bank can be defined with a sector granularity. Access to the hide protection area can be denied by progressing the HDPL level in the SBS.

When the HDPL = 1, no user flash is protected by the HDP. With HDPL ≥ 2 the user OB defined HDP area in each bank is closed. No read, write, fetch or erase is allowed in the HDP area.

The HDP area can be extended, the extension setting is only a register value, not stored in OB. With HDPL = 3 the HDP area remains activated and the extension is added. If extension is added while the base user OB defined area is not active, the extension covers one more sector, because HDPx_END sector is also protected.

The HDPL level can be only cleared by a system reset, there is no means to deactivate the HDP area.

The protected HDP area is defined by setting its size using start and end sectors in a similar way as the secure watermark.

The size of the inaccessible area can be extended using register FLASH_HDPEXTR values HDPx_EXT, representing the number of sectors added to the HDP area (past the HDPx_END sector). The volatile HDPx_EXT value can be increased, but cannot decrement.

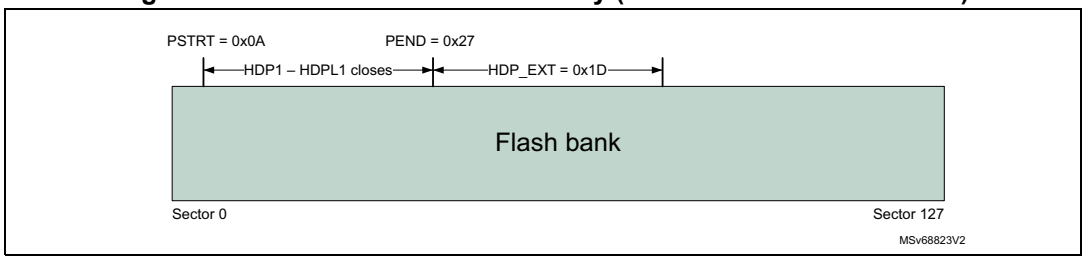
If HDPx_STRT > HDPx_END the extension covers (HDPx_EXT + 1) size area between sectors marked by HDPx_END and HDPx_END + HDPx_EXT.

By default HDPx_END is set to 0 and HDPx_STRT is set to 1. This means that the HDP area in user flash has a 0 size, and extends from the flash start address (first sector).

For example, to protect area by HDP from the address 0x0801 4000 (included) to the address 0x0804 FFFF (included):

- For physical Bank1, the option byte registers must be programmed with:
  - HDP1_STRT = 0x0A
  - HDP1_END = 0x27

Figure 33. HDP[a] in user flash memory (STM32H563/573xx devices)



- Once the HDPL is incremented to 2, the marked area becomes inaccessible. The HDPL = 2 code then sets the extension size to 29. Once the HDPL is incremented to 3, even the extension area becomes inaccessible.
- Alternatively, if:
  - HDP1_END = 0, HDP1_START = 1, HDPx_EXT = 1:
    Sector 0 and Sector 1 are HDP protected.
  - HDP1_END = 2, HDP1_START = 7, HDPx_EXT = 3 (a rather unusual example):
    Sectors 2 to 5 are HDP protected.
- If the two banks are swapped, the protection defined to physical Bank1 remains on the physical Bank1, unaffected by swapping. Separate protection applies to physical Bank2 and the option bytes registers must also be programmed with:
  - HDP2_STRT = 0x0A
  - HDP2_END = 0x27

Note: For more details on the bank swapping mechanism, refer to Section 7.6.6. The values have an upper limit, 127 for STM32H562/63/73xx devices, 31 for STM32H523/33xx devices.

Table 59. Secure hide protection

HDPx watermark option byte values (x = 1, 2)		Hide protection area
HDPL ≤ 1	-	No inaccessible HDP area in user flash memory
HDPL = 2	HDPx_END < HDPx_STRT	No inaccessible HDP area in user flash memory
	HDPx_END = HDPx_STRT	Exactly one sector is protected by HDP
	HDPx_END > HDPx_STRT	The area between HDPx_STRT and HDPx_END is HDP protected.
HDPL = 3	HDPx_END < HDPx_STRT and HDPx_EXT = 0	No inaccessible HDP area in user flash memory
	HDPx_END ≥ HDPx_STRT or HDPx_EXT > 0	The area between min (HDPx_STRT, HDPx_END) and (HDPx_END + HDPx_EXT) is HDP protected

a. The HDP works analogically, the number of sectors is the only differentiator.

Table 60. HDP protections summary

HDP protection	User flash			
	Access to OB HDP area		Access to EXT HDP area	
	HDPL 2 or 3	HDPL1	HDPL3	HDPL1 or HDPL 2
Fetch	BUS ERROR	OK	BUS ERROR	OK
Read	RAZ		RAZ	
Write	WI, WRPERR	If not WRP: OK Else: WI, WRPERR	WI, WRPERR	If not WRP: OK Else: WI, WRPERR
Erase (mass erase)				

7.6.3 Block-based secure flash memory area protection

Any sector can be programmed on-the-fly as secure or non-secure using the block-based configuration registers. FLASH_SECBB1x/FLASH_SECBB2x registers are used to configure the security attribute for, respectively, sectors in Bank1/Bank2.

When the sector security attribute bit SECBByx[i] is set, the security attribute is the same as the secure watermark-based area. The secure sector is accessible only by a secure access.

If the SECBBy[i] bit is set or reset for a sector already included in a secure watermark-based area, the sector keeps the watermark-based protection security attributes.

To modify a block-based sector security attribute, it is recommended to:

- Check that no flash operation is ongoing on the related sector.
- Add an ISB instruction after modifying the sector security attribute SECBByx[i].

Caution: Switching a sector from secure to non-secure does not erase the content of the associated sector. User secure software must perform the needed operations before switching to non-secure attribute:

- Erase sector content.
- Invalidate the instruction cache.

Note: For SECBByx[i] access control, refer to Table 61.

Execute protection for S-Bus (AHB register bus) is done in AHB decoder, not in flash memory interface itself.

Table 61. Secure configuration block-based registers access conditions

Access			Corresponding sector privilege status	Sector setting in SECBBxy
Fetch	Secure/ non-secure	Privilege/ unprivilege	-	BUS ERROR
Read			-	OK
Write	Secure	Privilege	-	OK
		Unprivilege	0	OK
			1	WI
	Non-secure	Privilege/ unprivilege	-	WI, ILAP

7.6.4 Block-based privileged flash memory area protection

Any sector can be programmed on-the-fly as privileged or unprivileged using the block-based configuration registers. FLASH_PRIVBB1x and FLASH_PRIVBB2x registers are used to configure the privilege attribute for, respectively, sectors in Bank1 and Bank2.

When the sector privilege attribute PRIVBByx[i] bit is set, the sector is accessible only by a privileged access. An unprivileged sector is accessible by a privileged or unprivileged access.

To modify a block-based privilege attribution, it is recommended to:

- Check that no flash operation is ongoing on the related sector.
- Add an ISB instruction after modifying the sector security attribute PRIVBByx[i].

Caution: Switching a sector from privileged to unprivileged does not erase the content of the associated sector.

Table 62. Privilege protection summary

Access (TZ not relevant)		Main flash	
		Unprivileged sector	Privileged sector
Privilege	Fetch	OK	
	Read		
	Write		
	Erase		
Unprivilege	Fetch	OK	RAZ
	Read		
	Write		WI, WRPERR
	Erase		

Table 63. Privilege and mass or bank erase

Access (TZ state not relevant)	Main flash		
	Unprivileged FLASH	Privileged FLASH	Mix unprivileged and privileged FLASH
MEP	OK		
MENP	OK	WI, WRPERR	

Note: For PRIVBByx[i] access control, refer to Table 64 and Table 65.

Table 64. Privilege configuration register access conditions (TZ enabled)

Access			Corresponding sector secure status(1)	Sector setting access in PRIVBBxy
Fetch	Privileged/ unprivileged	Secure/ non-secure	-	BUS ERROR
Read			-	OK (all sectors)

Table 64. Privilege configuration register access conditions (TZ enabled) (continued)

Access			Corresponding sector secure status(1)	Sector setting access in PRIVBBxy
Write	Privileged	Secure	-	OK (all sectors)
		Non-secure	NS	OK (only the sector corresponding to the bit)
			S	WI (only the sector corresponding to the bit)
	Unprivileged	Secure/ non-secure	-	WI

1. Either watermark or block-based.

Table 65. Privilege configuration register access conditions (TZ disabled)

Access		Sector setting access in PRIVBBRxy
Fetch	Privileged/ unprivileged	BUS ERROR
Read		OK
Write	Privileged	OK
	Unprivileged	WI

7.6.5 Flash memory register privileged and unprivileged modes

The flash registers can be read and written by privileged and unprivileged accesses depending on the SPRIV and NSPRIV bits in *FLASH privilege configuration register (FLASH_PRIVCFGR)*:

- When the SPRIV (respectively NSPRIV) bit is reset, all secure (respectively non-secure) flash registers can be read and written by both privileged or unprivileged access.
- When the SPRIV (respectively NSPRIV) bit is set, all secure (respectively non-secure) flash registers can be read and written by privileged access only. Unprivileged access to a privileged registers is RAZ/WI.

The registers related to key storage (FLASH_NSOBKCFGR, FLASH_SECOBKCFGR, FLASH_NSOBKKEYR, FLASH_SECOBKEYR) are exceptions to this rule and can be only accessed in privilege, regardless of other settings.

The next table summarizes flash registers access control.

Table 66. Flash register accesses(1)

Access			Non-secure register		Secure register	
			NSPRIV = 1	NSPRIV = 0	SPRIV = 1	SPRIV = 0
X	Secure/ non-secure	Privileged/ unprivileged	Bus error			
R/W	Secure	Privileged	OK			
R/W		Unprivileged	RAZ, WI	OK	RAZ, WI	OK

Table 66. Flash register accesses(1) (continued)

Access			Non-secure register		Secure register	
			NSPRIV = 1	NSPRIV = 0	SPRIV = 1	SPRIV = 0
R/W	Non-secure	Privileged	OK		RAZ, WI, ILAP	
R/W		Unprivileged	RAZ, WI	OK		

1. Some registers are privilege only, not configurable.

7.6.6 Flash memory banks attributes in case of bank swap

The SWAP_BANK option bit modifies the address of each bank in the memory map. When SWAP_BANK is reset, flash Bank1 is mapped at the lower address range. When SWAP_BANK is set, flash Bank1 is mapped at the higher address range. flash bank attributes follow their bank contents so there is no need to modify their setting registers when swapping banks:

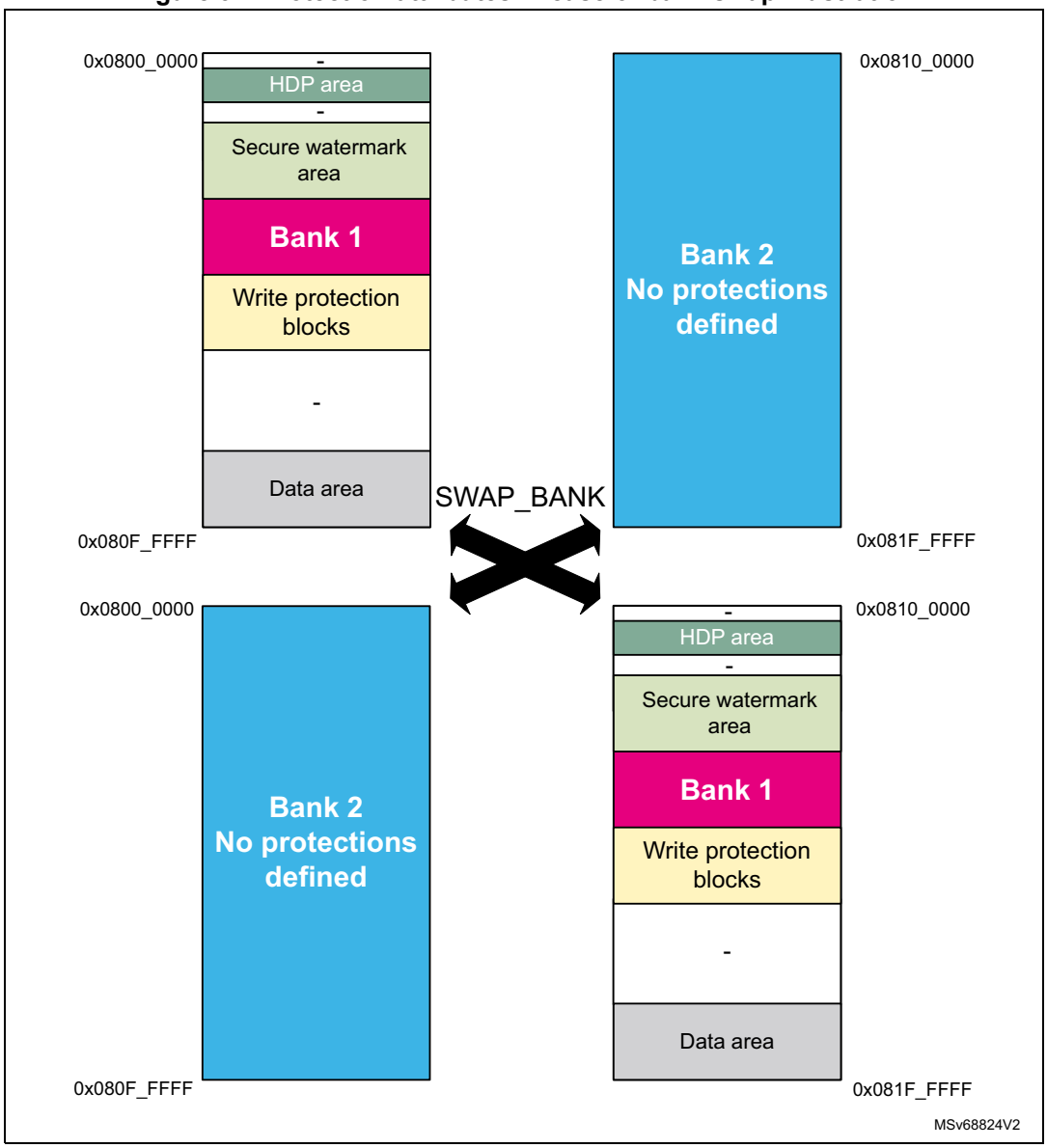
- Flash secure watermark x FLASH_SECWMx
- Flash write protection sector group FLASH_WRPSG (refer to *Section 7.6.8*)
- Hide protection in FLASH_HDPx registers
- Flash secure block based bank x register y FLASH_SECBBxy
- Flash privilege block based bank x register y FLASH_PRIVBBxy
- The data area configuration FLASH_EDATA

The SWAP_BANK is rendered immutable by setting BOOT_LOCK in the user OB. If TZEN is enabled, then SECBOOT_LOCK locks the option. If TZEN is disabled, then NSBOOT_LOCK locks the SWAP_BANK option.

Note: The BK_ECC bit in the FLASH ECC detection register (FLASH_ECCDETR) and FLASH ECC correction register (FLASH_ECCCORR), BKSEL bit in FLASH non-secure control register (FLASH_NSCR) and BKSEL bit in FLASH secure control register (FLASH_SECCR) always refers to Bank1 (respectively Bank2) when it is low (respectively high), regardless of the SWAP_BANK value.

Figure 34 shows how security attributes and protections behave in case of bank swap.

Figure 34. Protection attributes in case of bank swap illustration



7.6.7 Flash memory configuration protection

The memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (such as software bugs):

- Option bytes change
- Write operations
- Erase commands
- Interrupt masking

The memory configuration registers protection is summarized in *Table 67*. Registers not present in this table are not protected by a key.

Table 67. Flash interface register protection summary

Register name	Unlocking register	Protected asset
FLASH_NSCR	FLASH_NSKEYR	Write/erase control
FLASH_SECCR	FLASH_SECKEYR	Secure write/erase control
FLASH_OPTCR + all _PRG registers	FLASH_OPTKEYR	Flash bank option byte words change
FLASH_NSOBKCFGR	FLASH_NSOBKKEYR	Flash OBKey storage manipulation
FLASH_SECOBKCFGR	FLASH_SECOBKKEYR	

7.6.8 Write protection

The purpose of the write protection is to prevent unwanted modifications to code and/or data.

Any group of four consecutive 8-Kbyte sectors can be independently write-protected or unprotected by clearing/setting the corresponding WRPSGn1/2 bit in the FLASH_WRPSG1/2 register (see *FLASH write sector group protection for Bank1 (FLASH_WRP1R_CUR)* and *FLASH write sector group protection for Bank2 (FLASH_WRP2R_CUR)*).

A write-protected group of sectors cannot be erased nor programmed. As a result, a bank erase cannot be performed if one group of sectors is write-protected, unless a NVSTATE transition to OPEN is triggered (erasing the whole user flash memory).

Note: Write protection errors are documented in Section 7.9.

7.6.9 Flash high-cycle data protections

The memory can be configured to have 96-Kbyte memory area with high-cycling capability (100 kcycles) to store data and emulate EEPROM, see *Section 7.3.10*.

Option byte controlled security features for user flash memory prevent sectors from being used as high-cycle data area. No OPTCHANGEERR is raised in case of setting conflict, it only makes the sectors unusable. Details are listed in *Table 68*.

Volatile settings using FLASH_SECBBxx and FLASH_PRIVBBxx can be applied. In this case the security setting of user flash sector 127 applies to flash high-cycle data sector 0 and so on, as indicated in *Figure 25*, *Figure 26*, and *Figure 29*.

Volatile settings of HDP extension that extends into sectors of data area cause the data area to be inaccessible.

Table 68. High-cycle area protection summary: access to data area address range

<table>
<tr><th colspan="2" rowspan="3">TZ protection (when TZ_STATE = 0xB4)(1)</th><th colspan="6">Data sector enabled</th><th rowspan="3">Data sector not enabled</th></tr>
<tr><th rowspan="2">HDP protected</th><th rowspan="2">OB secure area</th><th colspan="2">SecBB protected</th><th colspan="2">NS sector</th></tr>
<tr><th>16/32 bits access</th><th>Other</th><th>16/32 bits access</th><th>Other</th></tr>
<tr><td rowspan="4">Secure</td><td>Fetch</td><td colspan="2" rowspan="3">BUS ERROR</td><td colspan="4">BUS ERROR</td><td rowspan="5">BUS ERROR</td></tr>
<tr><td>Read</td><td>OK</td><td rowspan="2">BUS ERROR</td><td colspan="2">RAZ, ILAC</td></tr>
<tr><td>Write</td><td>If not WRP: OK else WI, WRPERR</td><td colspan="2">WI, WRPERR, ILAFM</td></tr>
<tr><td>Erase</td><td colspan="6">Same protection as in corresponding classic user flash sector</td></tr>
<tr><td rowspan="4">Non-secure</td><td>Fetch</td><td colspan="2" rowspan="3">BUS ERROR</td><td colspan="4">BUS ERROR</td></tr>
<tr><td>Read</td><td colspan="2">RAZ, ILAC</td><td>OK</td><td rowspan="2">BUS ERROR</td></tr>
<tr><td>Write</td><td colspan="2">WI, WRPERR, ILAFM</td><td>If not WRP: OK else WI, WRPERR</td></tr>
<tr><td>Erase</td><td colspan="6">Same protection as in corresponding classic user flash sector</td></tr>
</table>

1. If TZ_STATE = 0xC3, there is no secure state access, no SecBB, and no Secure Area. For HDP and NS access the rules remain the same.

Table 69. HDP protected definition

User flash	OB HDP area enabled	HDP extension enabled
HDPL1	HDP protected = 0	HDP protected = 0
HDPL2	HDP protected = 1	HDP protected = 0
HDPL3	HDP protected = 1	HDP protected = 1

Table 70. Privileged sectors and data area - Access to data area address range

<table>
<tr><th colspan="2" rowspan="2">Access</th><th colspan="2">Data area address range access</th></tr>
<tr><th>Unprivileged sector</th><th>Privileged sector</th></tr>
<tr><td rowspan="3">Privileged</td><td>Fetch</td><td colspan="2">BUS ERROR</td></tr>
<tr><td>Read</td><td>OK</td><td>OK</td></tr>
<tr><td>Write</td><td>OK</td><td>OK</td></tr>
<tr><td rowspan="3">Unprivileged</td><td>Fetch</td><td colspan="2">BUS ERROR</td></tr>
<tr><td>Read</td><td>OK</td><td>RAZ, ILAC</td></tr>
<tr><td>Write</td><td>OK</td><td>WI, WRPERR</td></tr>
</table>

7.6.10 Life cycle management

Non-volatile or debug states are determined by the product state set by user option bytes. Debug possibility and possibility to change selected security settings are related to it. While the state is stored in the OB, it is the SBS that controls the debug policy.

OBK storage access conditions are unaffected by the PRODUCT_STATE, but keys can be evicted in regression transitions.

HDP works regardless of the debug state. Even in debug the HDP area is hidden when HDPL increments.

The following states are defined:

Table 71. Product states, debug states and debug policy

PRODUCT_STATE	Code in OB	Description
Open	0xED	User flash open (TZ secure and non-secure open). External access (debug) N and NS enabled (~RDP0).
Provisioning	0x17	Provisioning - Immutable root of trust is being installed. Open if TZEN = 0xC3. The debug is opened in HDPL3-NS.
iROT-Provisioned	0x2E	The immutable root of trust is installed (~RDP 0.5).
TZ-Closed	0xC6	State for debugging the NS application. Debug is restrained to non-secure areas.
Closed	0x72	State for running a secure application. Debug disabled, regression is possible.
Locked	0x5C	Transition to other state, policy change or debug not permitted(1).
Regression	0x9A	The temporary state initiated by the debug authentication system in transition to Open.
NS-Regression	0xA3	The temporary state initiated by the debug authentication system in transition to TZ-Closed.

1. The same situation is Closed state with incorrect or not defined debug certificates.

1. No debug protection

Read, program and erase operations into the flash main memory area are possible. Least restriction on the option bytes.

2. Non-secure debug only

All read and write operations (if no write protection is set) from/to the non-secure flash memory are possible. The debug access to secure area is prohibited. Debug access to non-secure area remains possible.

The following rules are enforced:

- User mode: code executing in user mode (boot flash) can access flash main memory and option bytes with all operations (read, erase, program).
- Non-secure debug mode: The secure flash memory areas are inaccessible; the non-secure flash memory areas remain accessible for debug purpose.
- Boot RAM mode: boot from SRAM is not possible.
- Transition to Open state is possible with the consequence of flash mass erase and key slots revocation. Depends on debug unlock policy (see below).

3. Full product protection

All debug features are disabled and following rules enforced:

- User mode: code executing in user mode (boot flash) can access flash main memory and selected option bytes with all operations (read, erase, program).
- Boot RAM mode: boot from SRAM is not possible.
- Transition to TZ-Closed or Open state is possible with the consequence of flash mass erase and key slots revocation. Depends on debug unlock policy (see below).

7.6.11 Product state transitions

Progressing towards a more closed state is the normal product life cycle, that does not require security measures. Transition towards Open state is a regression, controlled by the debug authentication control. If debug unlock policy is set to “locked”, no regression is accepted. Locked state can be reinforced by invalidating the debug authentication certificates.

All transitions not listed in this chapter are invalid.

There is no restriction on reading the current product state.

Transitions in progress direction

Transition from Open to any of the closed states is matter of correct product configuration and provisioning. The transitions must be done in correct order.

The normal progression is:

- Open to Provisioning or iRoT-Provisioned
- Provisioning to iRoT-Provisioned
- Provisioning or iRoT-Provisioned to TZ-Closed, Locked or Closed
- TZ-Closed to Locked or Closed

Transition is managed either by a software running on the device, or directly, using a debug interface. Transitions from Closed are only possible by software running on the device. By software it is assumed that transitions are triggered by bootloader, or root-of-trust services, but generally any software running on the device can do a progress transition.

Transition to Open state

This transition is a full regression. The starting state is any except Open and Locked. The debug tools are used to authenticate debug regression access rights with the debug authentication library, running on the device in HDPL1 (refer to *Section 14.3.6: SBS debug control*). After verifying the credentials, it puts the device into intermediate state Regression. From this state the device regresses securely in HDPL0 to Open.

The transition has the following consequence:

- Epoch secure counter is incremented
- Key slots revocation
- Product state updated
- User options updated (security settings reset, TZEN reset)
- Flash memory mass erase
- Backup RAM erase is requested

Transition from closed to closed-secure debug state

This transition opens the device to authorize the debug of the non-secure application software without compromising the security of the ROT functions.

The starting state is Closed. The debug tools are used to authenticate debug regression access rights with the debug authentication library, running on the device in the HDPL1 (refer to *Section 14.3.6: SBS debug control*). After verifying the credentials, it puts the device into intermediate state NS-Regression. From this state, the device regresses securely in HDPL0 to Closed.

The transition has the following consequence:

- Epoch non-secure counter is incremented
- Non-secure slots revocation
- Product state updated
- Flash memory sectors not covered by the secure watermark are erased
- Backup RAM erase is requested

Table 72. PRODUCT_STATE transitions

From	To(1)							
	Open	Provisioning	iROT-Provisioned	TZ-Closed(2)	Closed	Locked	Regression	NS-Regression(2)
Open	-	OK	OK	X	X	X	X	X
Provisioning	X	-	OK	OK	OK	OK	OK (HDPL1)	X
iRoT-Provisioned	X	X	-	OK	OK	OK	OK (HDPL1)	X
TZ-Closed	X	X	X	-	OK	OK	OK (HDPL1)	X
Closed	X	X	X	X	-	X	OK (HDPL1)	OK (HDPL1)
Locked	X	X	X	X	X	-	X	X

Table 72. PRODUCT_STATE transitions (continued)

From	To[1]							
	Open	Provisioning	iROT-Provisioned	TZ-Closed[2]	Closed	Locked	Regression	NS-Regression[2]
Regression	OK (HDPL0)	X	X	X	X	X	-	X
NS-Regression	X	X	X	OK (HDPL0)	X	X	X	-

1. X = transitions not permitted (OPTCHANGEERR raised), - = no change, OK = valid transitions.
2. To transition into this state, both the current and programmed values of TZEN must be enabled (TZEN = 0xB4), otherwise the change is ignored with OPTCHANGEERR.

In *Table 72*, X indicates transitions that are not permitted, a dash indicates no change, OK indicates valid transitions. Some transitions are possible only in correct HDPL provided in parenthesis. Attempts to do an illegal transition result in OPTCHANGEERR.

Incorrect PRODUCT_STATE (no OBL) is interpreted as Locked.

7.6.12 OBK protection

At the receipt of an OBK access (read/write/execute), secure and privilege attributes are first checked, followed by OBK-HDPL. The OBK-HDPL value must exactly match the HDPL assigned to the key location, otherwise OBKERROR flag is raised. Flash interface response is described in *Table 73*.

Table 73. TZ OBK protection summary

TZ protection (when TZ_STATE = 0xB4)	OBK access		
	No tamper		Tamper
	HDPL0/1/2/3	OBK selector	
XS	BUS ERROR	BUS ERROR	RAZ, WI
RS	OK if privilege, otherwise RAZ	RAZ	
WS	OK if privilege, otherwise WI, WRPERR	WI, if not privilege also WRPERR	
XNS	BUS ERROR	BUS ERROR	
RNS	RAZ, ILAFM	RAZ, ILAFM	
WNS	WI, WRPERR, ILAFM	WI, WRPERR, ILAFM	
Erase	NA	NA	NA

Table 74. OBK protection summary with TZ disabled

TZ protection (when TZ_STATE = 0xC3)	OBK access		
	No tamper		Tamper
	HDPL1/2/3	OBK selector	
XNS	BUS ERROR	BUS ERROR	RAZ, WI
RNS	OK if privileged, otherwise RAZ	RAZ	
WNS	OK if privilege, otherwise WI, WRPERR	WI if privileged, otherwise WI, WRPERR	
Erase	NA	NA	NA

Note: Cacheable access is managed at system level and not in the flash interface. The MPU must be configured to disable cache where it is not desirable.

In case of tamper, all keys are read as 0x00s.

There are also rules for accessing the OBKFGCR and OBKKEYR registers.

Table 75. Access conditions to secure control register

-	S/NS	P/NP	TZ state	SECOBKCFGR and SECOBKKEYR access
X	-	-	-	Bus error
R/W	S	P	Active	OK
R/W	S	NP	Active	RAZ, WI
R/W	NS	P	Active	RAZ, WI, ILAP
R/W	NS	NP	Active	RAZ, WI, ILAP
R/W	-	-	Deactivated	RAZ, WI

Table 76. Access conditions to non-secure control register

	S/NS	P/NP	TZ state	NSOBKCFGR and NSOBKKEYR access
X	-	-	-	Bus error
R/W	-	-	Active	RAZ, WI
R/W	-	P	Deactivated	OK
R/W	-	NP	Deactivated	RAZ, WI

7.6.13 One-time-programmable and read-only memory protections

Sections of OTP/RO in the flash memory are described in details in *Section 7.3.9*. There is no protection provided by the flash memory interface other than the dedicated write protection.

No write is possible to the RO area, once it was established in manufacturing. The *OTP write protection* describes a method of locking out the OTP area.

The access conditions are summarized in *Table 77*.

Table 77. OTP/RO access constraints

-	8 Kbytes sector dedicated to RO/OTP			
	0x08FF_E000 - E7FF	**0x08FF_E800 - EFFF Reserved**	**0x08FF_F000 - F7FF OTP**	**0x08FF_F800 - FFFF RO**
XS	BUS ERROR	BUS ERROR		
RS		RAZ, ILAC		
WS		WI, WRPERR, ILAFM		
XNS	BUS ERROR	BUS ERROR		
RNS		If correct size(1) RAZ, else BUS ERROR	If correct size OK, else BUS ERROR	
WNS		If correct size WI, else BUS ERROR	BUS ERROR if incorrect size. WRPERR if block is protected by LOCKBL, else OK.	If correct size WI, else BUS ERROR
Erase	WI			

1. Word size is 16 bits. 32-bit access is possible. Other access attempt leads to bus error.

7.7 System memory

7.7.1 Introduction

System memory stores RSS (root secure services) firmware programmed by ST during production. The RSS provides runtime services to user firmware.

When TrustZone is enabled (user sets TZEN bitfield to 0xB4) then RSS provides secure services to secure user firmware only; when TrustZone is disabled (user sets TZEN bitfield to 0xC3), RSS provides services to user firmware.

7.7.2 RSS user functions

The RSS provides runtime services thanks to RSS library, whose functions are exposed to user within the CMSIS device header file provided by the STM32CubeH5 firmware package (see UM3065 "*Getting started with STM32CubeH5 for STM32H5 Series*" for more details).

RSS provides services through two different libraries, RSSLIB and NSSLIB.

Table 78. RSSLIB/NSSLIB accesses

TZEN configuration	User FW Non-Secure	User FW Secure	Bootloader/JTAG/SWD(1)
Enabled (0xB4)	N/A	RSSLIB	RSSLIB
Disabled (0xC3)	NSSLIB	N/A	

1. When the devices boot in bootloader, that is, in product state OPEN (with BOOT pin set to 1 and UBE set to 0xB4), or PROVISIONING.

Table 79. RSSLIB/NSSLIB entry point access

C defined macro	Location in flash memory
RSSLIB_PFUNC	0xBF9FB68
NSSLIB_PFUNC	0xBF9FB6C

RSSLIB

The secure user firmware calls RSSLIB functions using RSSLIB_PFUNC C defined macro, which points to a location within non-secure system memory.Before calling RSSLIB functions, the secure user firmware must define a non-secure region above this location within SAU of the Cortex®-M33, starting from RSSLIB_SYS_FLASH_NS_PFUNC_START (0xBF9FB78), up to RSSLIB_SYS_FLASH_NS_PFUNC_END (0xBF9FB84). These last addresses are provided within the CMSIS device header file.

The user can set this non-secure region either by using the CMSIS system partition header file, or by implementing its own code for SAU setup. The CMSIS system partition header file is part of the STM32CubeH5 firmware package.

Table 80. RSS lib interface functions

Library	Function	Attributes
RSSLIB_PFUNC	*JumpHDPLvl2*	Secure callable function
	JumpHDPLvl3	
	JumpHDPLvl3NS	
	DataProvisioning	Non-secure callable function

DataProvisioning

Secure attribute: Non-secure callable function.

Prototype:

```
uint32_t_ DataProvisioning (RSSLIB_DataProvisioningConf_t *pConfig)
```

User code function call example:

```
RSSLIB_PFUNC→NSC.DataProvisioning(&pConfig)
```

Arguments:

pConfig: input parameter. **RSSLIB_DataProvisioningConf_t**

C structure definition is described below:

```
typedef struct
{
  uint32_t *pSource;
  uint32_t *pDestination;
  uint32_t Size;
  uint32_t DoEncryption;
  uint32_t Crc;
} RSSLIB_DataProvisioningConf_t;
```

Structure elements

pSource Provides the address of data to be provisioned. Must be within SRAM3 address range (non-secure aliases).

pDestination Provides the address where to store data to be provisioned. Must be within OBKeys address range. To be aligned on 16 bytes.

Size Provides the size of data to be provisioned (the number of bytes, must be a multiple of 16).

DoEncryption Notifies RSSLIB_DataProvisioning function if it must encrypt or not data within OBKeys.

DoEncrypt can be either 0xF5F5A0AAU or 0xCACA0AA0U (the only one allowed on STM32H563xx devices)

– 0xF5F5A0AAU: notifies RSSLIB_DataProvisioning to encrypt data with relevant DHUK before programming it within OBKeys. DHUK is selected according to **pDestination** value.

– 0xCACA0AA0U: notifies RSSLIB_DataProvisioning to program in clear data within OBKeys.

Crc CRC over full source data buffer, **pConfig->pDestination** value, **pConfig->Size** value and finally **pConfig->DoEncryption** value.

CRC computation uses CRC-32 (Ethernet):

– CRC polynomial: 0x04C11DB7U

– Initial value: 0xFFFFFFFFU

Returned values

0xEAEAEAEAU	Success
0xF5F5E0E0U	Error: – **pConfig** or **pSource** are not in SRAM3 – **pDestination** is not aligned on 16 bytes – **pDestination + Size** does not fit within an OBKey section
0xF5F50E0EU	Error: **Size** is not a multiple of 16
0xF5F58080U	Error: computed CRC is not the expected one provided within **pConfig->CRC**
0xF5F58008U	Error: cannot program data within OBKeys destination section
0xF5F5E00EU	Error: wrong **DoEncryption** parameter value
0xF5F50880U	Error: encryption requested, butw platform does not support it
0xF5F50EE0U	Error: encryption error
0xF5F50808U.	Error: OBKeys programming error

RSSLIB_DataProvisioning receives in input a data buffer and programs it within OBKeys. A CRC prevents any data and parameters tampering issue.

The destination address (**pDestination**) added to the size (**Size**) of the data to be provisioned must not cross the OBKeys section boundaries, listed below.

OBKeys Level1	Start address: 0x0FFD0100UL
	End address: 0x0FFD08FFUL

(continued)

OBKeys Level2	Start address: 0x0FFD0900UL
	End address: 0x0FFD0BFFUL
OBKeys Level3 secure	Start address: 0x0FFD0C00UL
	End address: 0x0FFD17FFUL
OBKeys Level3 non-secure	Start address: 0x0FFD1800UL
	End address: 0x0FFD1FEFUL

When requested through **pConfig->DoEncryption** parameter, RSSLIB_DataProvisioning encrypts data before programming them within OBKeys.

RSSLIB_DataProvisioning uses AES CBC 128 bits with:

- IV: defined as (using C definition format)

  ```
  uint32_t IV = {0x8001D1CEU, 0xD1CED1CEU, 0xD1CE8001U, 0xCED1CED1U};
  ```

- Key: DHUK corresponding to the targeted OBKeys section defined by **pConfig->pDestination** parameter.

Note: Data encryption within OBkeys is supported only by STM32H533/573xx devices.

JumpHDPLvl2

Secure attribute: Secure callable function.

Prototype:

```
uint32_t JumpHDPLvl2(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
RSSLIB_PFUNC->S.JumpHDPLvl2((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.
- MPUIndex:
  - Input parameter, MPU region index. Caller function must define (but keep disabled) the corresponding MPU region before calling JumpHDPLvl2. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

User calls JumpHDPLvl2 to close user Flash HDPL1 area by incrementing HDPL to 2, then jump to the reset handler embedded within the vector table, whose address is passed as input parameter.

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by it. JumpHDPLvl2 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), RSSLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

JumpHDPLvl3

Secure attribute: Secure callable function.

Prototype:

```
uint32_t JumpHDPLvl3(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
RSSLIB_PFUNC->S.JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.
- MPUIndex:
  - Input parameter, MPU region index. Caller function must define but keep disabled the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table, whose function belongs to the MPU region.

User calls JumpHDPLvl3 to close user Flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3, then jump to the reset handler embedded within the vector table, whose address is passed as input parameter.

After closing HDPL1/2, JumpHDPLvl3 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by the vector table. JumpHDPLvl3 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), JumpHDPLvl3 returns 0xF5F5F5F5.

JumpHDPLvl3NS

Secure attribute: Secure callable function.

Prototype:

```
uint32_t JumpHDPLvl3NS(uint32_t VectorTableAddr)
```

User code function call example:

```
RSSLIB_PFUNC->S.JumpHDPLvl3NS((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.

User calls JumpHDPLvl3NS to close user flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3, to move from secure to non-secure domain, then jump to the non-secure reset handler embedded within the vector table, whose address is passed as input parameter.

After closing HDPL1/2, JumpHDPLvl3 jumps to the non-secure reset handler function supported by the vector table. JumpHDPLvl3NS does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), JumpHDPLvl3NS returns 0xF5F5F5F5.

NSSLIB

The user firmware calls only the NSSLIB function when TrustZone is disabled (TZEN bitfield is set to 0xC3).

The user firmware calls NSSLIB functions using NSSLIB_PFUNC C defined macro, which points to a location within system memory.

Table 81. NSS lib interface functions

Library	Function	Attributes
NSSLIB_PFUNC	JumpHDPLvl2	Non-secure function
	JumpHDPLvl3	

JumpHDPLvl2

Prototype:

```
uint32_t JumpHDPLvl2(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
NSSLIB_PFUNC->JumpHDPLvl2((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.
- MPUIndex:
  - Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl2. The function enables

the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

User calls JumpHDPLvl2 to close user Flash HDPL1 area by incrementing HDPL to 2, and to jump to the reset handler embedded within the vector table which address is passed as input parameter

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table and jumps to the reset handler function supported by the vector table too. JumpHDPLvl2 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), RSSLIB_Sec_JumpHDPLvl2 returns 0xF5F5F5F5.

JumpHDPLvl3

Prototype:

```
uint32_t JumpHDPLvl3(uint32_t VectorTableAddr, uint32_t MPUIndex)
```

User code function call example:

```
NSSLIB_PFUNC->JumpHDPLvl3((uint32_t)NextVectorTableAddr, 1U );
```

Arguments:

- VectorTableAddr:
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M33 core.
- MPUIndex:
  - Input parameter, MPU region index. Caller function shall define but keep disable the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

User calls JumpHDPLvl3 to close user Flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3, and then jump to the reset handler embedded within the vector table, whose address is passed as input parameter

After closing HDPL1/2, JumpHDPLvl3 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by it. JumpHDPLvl3 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), JumpHDPLvl3 returns 0xF5F5F5F5.

7.8 FLASH low-power modes

Table 82 summarizes the memory behavior in STM32 low-power modes. Embedded flash memory belongs to the core domain.

Table 82. Effect of low-power modes on the embedded flash memory

Power mode	Core domain voltage range	Allowed if FLASH busy	FLASH power mode
Run	VOS0/1/2/3	Yes	Run
Stop1 (clock stopped)	SVOS3/4/5	No	Clock gated or stopped in case of SVOS5
Standby	Off	No	Off

When the system state changes or within a given system state, the memory can operate with a different voltage supply range (VOS), according to the application. The procedure to switch the memory into power modes (run, clock gated, stopped, off) is described hereafter.

Note: For more information on the microcontroller power states, refer to Section 10: Power control (PWR).

Managing the FLASH domain switching to Stop or Standby

As explained in *Table 82*, if the memory informs the reset and clock controller (RCC) that it is busy (BSY, DBNE, WBNE is set), the microcontroller cannot switch the core domain to Stop or Standby mode.

There are two ways to release the memory:

- Reset the WBNE busy flag in FLASH_NS/SECSR register by any of the following actions:
  - Complete the write buffer with missing data.
  - Force the write operation without filling the missing data by activating the FW bit in FLASH_NS/SECCR register. This forces all missing data “high”.
- Poll BSY busy bits in FLASH_NS/SECSR register until they are cleared. This indicates that all recorded write, erase and option change operations are complete.

The microcontroller can then switch the domain to Stop or Standby mode.

7.9 FLASH error management

7.9.1 Introduction

The memory automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- *Non-secure write protection error (WRPERR)*
- *Secure write protection error (WRPERR)*
- *Non secure programming sequence error (PGSERR)*
- *Secure programming sequence error (PGSERR)*
- *Secure strobe error (STRBERR)*
- *Error correction code error (ECCC, ECCD)*
- *Illegal access (ILAFM/ILAP)*
- *Option byte change error (OPTCHANGEERR)*
- OBK non-secure general error (OBKERR)
- OBK secure general error (OBKERR)
- OBK non-secure write error (OBKWERR)
- OBK secure write error (OBKWERR)

The application software can individually enable the interrupt for each error, as detailed in *Section 7.10*.

The flash memory interface uses different interrupt lines to trigger the event. There are two direct lines to NVIC, one for secure interrupts, other for non secure. Third line leads to TZIC, for the ILAFM interrupt.

Note: For all errors, the application software must clear the error flag before attempting a new modify operation.

Since there is just one write buffer and only one operation is allowed at a time, the write control bits and status bits are shared by both banks:

- Write control bits (PG, FW, EOPIE, WRPERRIE, PGSERRIE, STRBERRIE, INCERRIE, CLR_EOP, CLR_WRPERR, CLR_STRBERR, CLR_INCERR, CLR_PGSERR) controls Bank1 and 2 at the same time.
- Write status flag reports (WBNE, DBNE, EOP, WRPERR, PGSERR, STRBERR, INCERR, BSY) error for Bank1 and 2 at the same time.

7.9.2 Non-secure write protection error (WRPERR)

When an illegal non-secure erase/program operation is attempted to the nonvolatile memory, the flash interface sets the write protection error flag WRPERR in FLASH_NSSR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A sector write-protected with WRPSGn
- An HDP area while the HDPL has made it inaccessible
- Secure (TZ) state not matching the erased memory attribute
- Attempt to erase privilege sector within unprivileged mode

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- System flash memory
- A user sector write-protected with WRPSGn
- An OTP block, locked with LOCKBL
- A read-only section
- A reserved area
- Secure area
- Privileged area from within unprivileged mode
- OBK access conditions check error

When WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank. If this error is detected, the write buffer is invalidated.

Note: WRPERR flag must be cleared before any erase/program operation.

WRPERR flag is cleared by setting CLR_WRPERR bit to 1 in FLASH_NSCCR register.

If WRPERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when WRPERR flag is raised (see *Section 7.10* for details).

7.9.3 Secure write protection error (WRPERR)

When an illegal secure erase/program operation is attempted to the nonvolatile memory, the flash interface sets the write protection error flag WRPERR in FLASH_SECSR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- A sector write-protected with WRPSGn
- An HDP area while the HDPL has made it inaccessible
- Secure (TZ) state not matching the erased memory attribute
- Attempt to erase privilege sector within unprivileged mode

A program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- System flash memory
- A user sector write-protected with WRPSGn
- An OTP block, locked with LOCKBL
- A read-only section
- A reserved area
- Non-secure area
- Privileged area from within unprivileged mode
- OBK access condition check error

When WRPERR flag is raised, the operation is rejected and nothing is changed in the corresponding bank.

If this error is detected the write buffer is invalidated.

Note: WRPERR flag must be cleared before any erase/program operation.

WRPERR flag is cleared by setting CLR_WRPERR bit to 1 in FLASH_SECCCR register.

If WRPERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when WRPERR flag is raised (see *Section 7.10* for details).

7.9.4 Non secure programming sequence error (PGSERR)

When the non-secure programming sequence is incorrect, the flash interface sets the programming sequence error flag PGSERR in FLASH_NSSR register.

More specifically, PGSERR flag is set when one of below conditions is met:

- An error like INCERR, WRPERR, PGSERR, STRBERR, OPTCHANGEERR, OBKERR or OBKWERR have not been cleared before requesting a new write or erase operation, OBK operation or options change.
- For erase, PGSERR is set if:
  - missing erase operation (FLASH_NSCR): STRT = 1 with (MER = 0, SER = 0 and BER = 0)
  - sector and bank erase requested at the same time (NSSTRT and more than one of MER, SER and BER)
  - PG set during erase operation (it ensures that no erase req and write req happen at the same time): (STRT = 1) with (PG = 1)
  - Erase operation is started while write buffer is waiting for next data: (STRT = 1) with WBNE = 1.
  - STRT = 1 set by secure access.
  - Erase operation is started while DBNE = 1.
- For programming, PGSERR is set if:
  - missing write flag: A write operation is requested but the program enable bit PG has not been set in FLASH_NSCR register prior to the request.
  - write operation to flash high-cycle data is requested but PG = 0.
  - AHB write request is received and (SER = 1, BER = 1 or MER = 1)
  - 16 bit data access requested while WBNE = 1
- For options change, PGSERR is set if:
  - option change is started while write buffer is waiting for next data: OPTSTRT = 1 with WBNE = 1
  - OPTSTRT set with non-secure DBNE = 1.
- For options byte key storage access, PGSERR is set if:
  - error flags from previous operation not cleared
  - ALT_SECT_ERASE and SWAP_SECT_REQ in FLASH_NSOBKCFGR are set at the same time
  - SWAP_OFFSET value is wrong: when SWAP_SECT_REQ is set to 1, the SWAP_OFFSET should be equal or greater than OBK_HDPL< N - 1 > _SWAP_OFFSET
  - ALT_SECT_ERASE or OBK_SWAP set while DBNE = 1
  - OBK-HDPL is incorrect and non-secure OBK_SWAP is requested
  - If OBKSWAP or OBK alternate erase is started while write buffer is waiting for new data (WBNE = 1).

When PGSERR flag is raised as consequence of failed write attempt (flash programming), the current program operation is aborted and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

Note: When PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. It is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful, by reading back the value in the flash memory.

The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

PGSERR flag is cleared by setting CLR_PGSERR bit to 1 in FLASH_NSCCR register.

If PGSERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when PGSERR flag is raised. See *Section 7.10* for details.

7.9.5 Secure programming sequence error (PGSERR)

When the secure programming sequence is incorrect, the flash interface sets the programming sequence error flag PGSERR in FLASH_SECSR register.

More specifically, PGSERR flag is set when one of below conditions is met:

- An error like INCERR, WRPERR, PGSERR, STRBERR, OBKERR, or OBKWERR has not been cleared before requesting a new write or erase operation, OBK operation or options change.
- For erase, PGSERR is set if:
  - missing erase operation (FLASH_SECCR): STRT = 1 with (SER = 0 and BER = 0, MER = 0)
  - sector and bank erase requested at the same time (SECSTRT and more than one of MER, SER, and BER)
  - PG set during erase operation (it ensures that no erase request and write request happen at the same time): (STRT = 1) with (PG = 1)
  - erase operation is started while write buffer is waiting for next data: (STRT = 1) with WBNE = 1.
  - operation started while secure DBNE = 1.
- For programming, PGSERR is set if:
  - missing write flag: a write operation is requested but the program enable bit PG has not been set in FLASH_SECCR register prior to the request
  - write operation to flash high-cycle data is requested but PG = 0
  - AHB write request is received and (SER = 1, SBER = 1, or MER = 1)
  - 16 bit data access requested while WBNE = 1
- For options change, PGSERR is set if:
  - Option change is started while write buffer is waiting for next data: OPTSTRT = 1 with WBNE = 1
  - OPTSTRT set with secure DBNE = 1
- For options byte key storage access, SECPGSERR is set if:
  - error flags from previous operation not cleared
  - ALT_SECT_ERASE and SWAP_SECT_REQ are set at the same time
  - SWAP_OFFSET value is wrong: when SWAP_SECT_REQ = 1, SWAP_OFFSET must be equal or greater than OBK_HDPL<N-1>_SWAP_OFFSET.
  - SWAP_SECT_REQ or OBK_SWAP set while DBNE = 1
  - OBK-HDPL is incorrect and secure OBK_SWAP is requested

– If OBKSWAP or OBK alternate erase is started while write buffer is waiting for new data (WBNE = 1)

When PGSERR flag is raised as consequence of failed write attempt (flash programming), the current program operation is aborted and nothing is changed in the corresponding bank. The write data buffer is also invalidated.

Note: When PGSERR flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. It is recommended to generate interrupts on PGSERR and verify in the interrupt handler if the last write operation has been successful by reading back the value in the flash memory.

The PGSERR flag also blocks any new program operation. This means that PGSERR must be cleared before starting a new program operation.

PGSERR flag is cleared by setting CLR_SECPGSERR bit to 1 in FLASH_SECCCR register.

If PGSERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when PGSERR flag is raised. See *Section 7.10* for details.

7.9.6 Non-secure strobe error (STRBERR)

When the non-secure application software writes several times to the same byte in the write buffer, the flash interface sets the strobe error flag STRBERR (FLASH_NSSR) whatever the target bank of the write access.

When STRBERR flag is raised, the current program operation is aborted and the write buffer is invalidated.

STRBERR flag is cleared by setting CLR_STRBERR bit to 1 in FLASH_NSCCR register.

If STRBERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when STRBERR flag is raised. See *Section 7.10* for details.

7.9.7 Secure strobe error (STRBERR)

When the secure application software writes several times to the same byte in the write buffer, the memory sets the strobe error flag STRBERR (FLASH_SECSR) whatever the target bank of the write access.

When STRBERR flag is raised, the current program operation is aborted and the write buffer is invalidated.

STRBERR flag is cleared by setting CLR_STRBERR bit to 1 in FLASH_SECCCR register.

If STRBERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when STRBERR flag is raised. See *Section 7.10* for details.

7.9.8 Non-secure inconsistency error (INCERR)

When a programming inconsistency in non-secure access is detected, the flash interface sets the inconsistency error flag INCERR in register FLASH_NSSR.

More specifically, INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.
  - ALT_SECT in FLASH_NSOBKCFGR changed while filling the write buffer.

Note: INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow the following sequence to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when INCERR flag is raised.
2. Stop all write requests to the memory.
3. Clear INCERR bit.
4. Restart the write operations from where they have been interrupted.

INCERR flag is cleared by setting CLR_INCERR bit to 1 in FLASH_NSCCR register.

If INCERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when INCERR flag is raised (see *Section 7.10* for details).

7.9.9 Secure inconsistency error (INCERR)

When a programming inconsistency is detected, the flash interface sets the inconsistency error flag INCERR in register FLASH_SECSR.

More specifically, INCERR flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.

Note: INCERR flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERR) is raised.

It is recommended to follow the sequence below to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when INCERR flag is raised.
2. Stop all write requests to the flash memory.
3. Clear INCERR bit.
4. Restart the write operations from where they have been interrupted.

INCERR flag is cleared by setting CLR_INCERR bit to 1 in FLASH_SECCCR register.

If INCERRIE bit in FLASH_SECCR register is set to 1, an interrupt is generated when INCERR flag is raised (see *Section 7.10* for details).

7.9.10 Error correction code error (ECCC, ECCD)

When a single-error correction is detected during a read, the flash interface sets the single-error correction flag ECCC in FLASH_ECCCORR register.

When two ECC errors are detected during a read, the flash interface sets the double error detection flag ECCD in FLASH_ECCDETR register.

When the ECCC flag is raised, the corrected read data are returned. The application can ignore the error, and request new read operations. When the ECCD the flag is raised, an NMI is generated, it can be masked in SBS registers (*SBS flift ECC NMI mask register (SBS_ECCNMIR)*) for data access (OTP, data area, RO data). Software must invalidate the instruction cache (CACHEINV = 1) in the NMI interrupt service routine when the ECCD flag is set.

When ECCC or ECCD flag is raised, the address of the flash word that generated the error is saved in the FLASH_ECCCORR (FLASH_ECCDETR) register. If the address corresponds to a read-only area or to an OTP area or flash high-cycle data, the OTP_ECC bit is also set to 1 in the FLASH_ECCCORRR (FLASH_ECCDETR) register. This register is automatically cleared when the associated flag that generated the error is reset.

A BK_ECC flag indicates in which flash bank the error occurred.

A SYSF_ECC flag indicates an error detected in the system flash area.

Table 83. Locating ECC failure

OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Flash area	ADDR_ECC min	ADDR_ECC max
0	0	0/1	0	0	User flash	0x0000	0xFFFF
0	1	0/1	0	0	System flash	0x0000	0x0FFF
1	0	0	0	0	OTP	0x0600	0x07FF
0	0	1	0	1	OBKeys	0x0000	0x03FF
0	0	0/1	1	0	Data area sector 7	0xF000	0xF1FF
0	0	0/1	1	0	Data area sector 6	0xF200	0xF3FF
0	0	0/1	1	0	Data area sector 5	0xF400	0xF5FF
0	0	0/1	1	0	Data area sector 4	0xF600	0xF7FF
0	0	0/1	1	0	Data area sector 3	0xF800	0xF9FF
0	0	0/1	1	0	Data area sector 2	0xFA00	0xFBFF
0	0	0/1	1	0	Data area sector 1	0xFC00	0xFDFF
0	0	0/1	1	0	Data area sector 0	0xFE00	0xFFFF

Note: In case of successive single correction or double detection errors, only the address corresponding to the first error is stored in FLASH_ECCCORR (FLASH_ECCDETR) *register.*

It is mandatory to clear ECCC or ECCD flags before starting a new read operation.

As the ECC interface is shared by the two banks, if the same error is registered simultaneously, a physical Bank1 error is registered. Both errors are registered if one bank reports ECCD and the other bank ECCC.

ECCC (respectively ECCD) flag is cleared by setting to 1 ECCC bit (respectively ECCD bit) in FLASH_ECCCORR (FLASH_ECCDETR) register.

If ECCC bit in FLASH_ECCCORR register is set to 1, an interrupt is generated when ECCC flag is raised. Only NMI is generated for the ECCD. See *Section 7.10* for details.

7.9.11 Illegal access (ILAFM/ILAP)

Illegal access is a signal to the TZIC, triggering a secure interrupt in there. It is complementary to the other interrupts and only generated when the TZ is enabled (TZ_STATE = 0xB4).

It can be masked only on GTZC level.

ILAFM is generated when rules on secure flash memory access are violated:

- Attempt to access secure memory location in non-secure mode.
- Attempt to access non-secure memory location in secure mode.

ILAP is generated on:

- Attempt to access secure register in non-secure mode.

If ILAFM is detected during write, the write buffer is invalidated.

Note: ILAFM and ILAP has no flag within the flash memory controller to clear to resume operation. It is dealt within the TZIC.

7.9.12 Option byte change error (OPTCHANGEERR)

When the flash interface finds an error during an option change operation, it aborts the operation and sets the option byte change error flag OPTCHANGEERR in FLASH_NSSR register.

The error is raised after OPTSTRT bit set if:

- TZEN is not set and transition to TZ-Closed or NS-Regression state is requested. Also any other forbidden PRODUCT_STATE transition (*Table 72*).
- SWAP_BANK is locked out by combination of TZEN and BOOT_LOCK.
- OB modification is attempted in wrong HDPL (selected PRODUCT_STATE transitions)
- Attempt to modify OB with invalid value
  - New EPOCH value not greater than current
  - Value not recognized by the specification (enumerated magic numbers only)
- Attempt to modify OB in product state where it is not allowed (usually a state open for debug is required for change).

Note: Exceptions and details provided in the OB description (see Section 7.4.7 and Table 52).

OPTCHANGEERR flag is cleared by setting CLR_OPTCHANGEERR bit to 1 in FLASH_NSCCR register.

If OPTCHANGEERRIE bit in FLASH_NSCR register is set to 1, an interrupt is generated when OPTCHANGEERR flag is raised (see *Section 7.10* for details).

It is mandatory to clean the OPTCHANGEERR flag before starting a new operation (option change, erase or write).

7.9.13 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On main AHB system bus for access targeting code and data with 9 bits ECC.
  - Fetching from secure user flash memory in non-secure mode.
  - Fetching from non-secure user flash memory in secure mode.
  - Access to invalid address (including data addresses forbidden for code use).
  - Fetching from HDP area with incorrect HDPL value.
  - Fetching from privilege area in unprivilege mode.
- On AHB configuration or system bus for accesses targeting OTP/RO (all addresses using 6bits ECC):
  - wrong key input to FLASH_NS/SECKEYR or FLASH_OPTKEYR.
  - 8-bit accesses to system AHB interface.
  - Wrong unlock sequence on a register.

7.9.14 OBK error cases (OBKERR, OBKWERR)

Four error types can be detected while filling the OBK write buffer: STRBERR, INCERR, WRPERR or OBKERR. The OBKERR is specific to the OBK access and cannot be raised by access to any other flash memory asset.

Note: For STRBERR, INCERR and WRPERR, non-secure or secure flags are used depending on the AHB access type (sec or non-secure). If a non-secure access is received, the non-secure error flags are used. If a secure access is received, the secure error flags are used.

For OBKERR and OBKWERR, non-secure or secure flags are used depending on the value of the secure state. If the SOC is in secure state (TZ_STATE active), the secure error flags (OBKERR and OBKWERR) are used. Otherwise the non-secure error flags (OBKERR and OBKWERR) are used.

OBKERR is raised when:

- the OBK-HDPL (input signal from SBS) does not match the HDPL associated to the key during a read or write access, and the TZ and PRIV attributes are correct (See *Section 7.5.2* for more details).

If OBKERR is raised during write access, the write buffer is flushed. If the OBKERR is raised on read access, the write buffer is intact. In both cases the flag must be cleared to enable write/erase access to flash memory.

OBKWERR is detected after filling the write buffer and raised if:

- the address is not virgin during a write access.
- OBK selector in the alternate sector is not virgin during a SWAP operation.

If OBKWERR is raised, the write buffer is flushed. The flag must be cleared to enable write/erase access to flash memory.

7.10 FLASH interrupts

The flash interface can generate a maskable interrupt to signal the following events on a given bank:

- Read and write errors (see *Section 7.9*)
  - Single ECC error correction during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - option change operation error
- Security errors (see *Section 7.9*)
  - Write protection error
  - Illegal access error - signal to TZIC
- Miscellaneous events (described below)
  - End of programming

A NMI is raised on double ECC error detection during read operation.

Two interrupt lines gather all the error sources, one for errors in non-secure access and second for errors that happen in secure access.

The user can individually enable or disable flash interface interrupt sources by changing the mask bits in the FLASH_NS/SECCR, FLASH_ECCCORR and FLASH_ECCDETR register. Setting the appropriate mask bit to 1 enables the interrupt.

Note: Prior to writing, FLASH_NS/SECCR register must be unlocked as explained in Section 7.6.7.

Table 84 gives a summary of the available flash interface interrupt features. As mentioned in the table below, some flags need to be cleared before a new operation is triggered.

Table 84. Flash interrupt request

Interrupt event Event flag	Error flag label in register	Enable control bit	Clear flag to resume operation
End of operation event	EOP	EOPIE	N/A
Write protection error	WRPERR	WRPERRIE	Yes
Programming sequence error	PGSERR	PGSERRIE	Yes
Strobe error	STRBERR	STRBERRIE	Yes
OBK write error	OBKWERR	OBKWERRIE	Yes
OBK general error	OBKERR	OBKERRIE	Yes
Inconsistency error	INCERR	INCERRIE	Yes
Illegal access error	ILAFM	TZEN	No
Option byte error	OPTCHANGEERR	-	Yes

Table 84. Flash interrupt request (continued)

Interrupt event Event flag	Error flag label in register	Enable control bit	Clear flag to resume operation
ECC single error correction event	ECCC	ECCCIE	No
ECC double error detection event(1)	ECCD	Disabled	No

1. NMI.

The status of the individual maskable interrupt sources described in *Table 84* (except for option byte error and ECC) can be read from the FLASH_NS/SECSR register. They can be cleared by setting to 1 the adequate bit in FLASH_NS/SECCCR register.

Note: No unlocking mechanism is required to clear an interrupt.

End of operation event

Setting the end of operation interrupt enable bit (EOPIE) in the FLASH_NS/SECCR register enables the generation of an interrupt at the end of an erase operation, a program operation, or an option byte change.

When managing the OBKey storage area, the EOP is associated also with the OBK swap operation and the alternate area erase operation. EOP bit in the FLASH_NS/SECSR register is also set when one of these events occurs.

Setting CLR_EOP bit to 1 in FLASH_NS/SECCCR register clears EOP flag.

7.11 FLASH registers

Each register has an offset address and a reset value. In case of registers representing option bytes, the reset value is determined by the OBL process. In case of success the reset value is loaded from OB, in case of failure, a highly restrictive default value is set.

7.11.1 FLASH access control register (FLASH_ACR)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

For more details, refer to *Section 7.3.4* and *Section 7.3.5*.

Address offset: 0x000

Reset value: 0x0000 0013

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PRFTEN**: Prefetch enable. When bit value is modified, user must read back ACR register to be sure PRFTEN has been taken into account.
Bits used to control the prefetch.
0: prefetch disabled.
1: prefetch enabled when latency is at least one wait state.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **WRHIGHFREQ[1:0]**: Flash signal delay
These bits are used to control the delay between nonvolatile memory signals during programming operations. Application software has to program them to the correct value depending on the memory interface frequency. Please refer to *Table 45* for details.
Note: No check is performed to verify that the configuration is correct.
Two WRHIGHFREQ values can be selected for some frequencies.

Bits 3:0 **LATENCY[3:0]**: Read latency
These bits are used to control the number of wait states used during read operations on both nonvolatile memory banks. The application software has to program them to the correct value depending on the memory interface frequency and voltage conditions.
0000: no wait states used to read a word from nonvolatile memory
0001: one wait state used to read a word from nonvolatile memory
0010: two wait states used to read a word from nonvolatile memory
...
0111: seven wait states used to read a word from nonvolatile memory
1111: 15 wait states used to read a word from nonvolatile memory
Note: No check is performed by hardware to verify that the configuration is correct.

7.11.2 FLASH non-secure key register (FLASH_NSKEYR)

This register is non-secure. It can written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_NSKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_NSCR register and allow programming/erasing it.

- First key = 0x4567 0123
- Second key = 0xCDEF 89AB

A wrong sequence locks the FLASH_NSCR register until next system reset.

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:0 **NSKEY[31:0]**: Non-volatile memory non-secure configuration access unlock key

7.11.3 FLASH secure key register (FLASH_SECKEYR)

This register is secure. It can be written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

FLASH_SECKEYR is a write-only register. The following values must be programmed consecutively to unlock FLASH_SECCR register and allow programming/erasing it. A wrong sequence locks the FLASH_SECCR register until next system reset.

First key = 0x4567 0123

Second key = 0xCDEF 89AB

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:0 **SECKEY[31:0]**: Non-volatile memory secure configuration access unlock key

7.11.4 FLASH option key register (FLASH_OPTKEYR)

This write-only register is non-secure. It can be written by both secure and non-secure accesses, and can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

The following values must be programmed consecutively to unlock FLASH_OPTCR register

- First key = 0x0819 2A3B
- Second key = 0x4C5D 6E7F

A wrong sequence locks FLASH_OPTCR register until next system reset.

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:0 **OPTKEY[31:0]**: FLASH option bytes control access unlock key

7.11.5 FLASH non-secure OBK key register (FLASH_NSOBKKEYR)

This is a write-only register. The following values must be programmed consecutively to unlock it register (a wrong sequence locks the register until next system reset).

- First key = 0x192A 083B
- Second key = 0x6E7F 4C5D

This register is non-secure. This register can be written only by privileged access.

Address offset: 0x0010

Reset value: 0x0000 0000

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

Bits 31:0 **NSOBKKEY[31:0]**: FLASH non-secure option bytes keys control access unlock key

7.11.6 FLASH secure OBK key register (FLASH_SECOBKKEYR)

This is a write-only register. The following values must be programmed consecutively to unlock it (a wrong sequence locks the register until the next system reset)

- First key = 0x192A 083B
- Second key = 0x6E7F 4C5D

This register is secure, it can be written only by privileged access.

Address offset: 0x0014

Reset value: 0x0000 0000

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

Bits 31:0 **SECOBKKEY[31:0]**: FLASH secure option bytes keys control access unlock key

7.11.7 FLASH operation status register (FLASH_OPSR)

This register is non-secure. This register can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x0018

Reset value: 0xXXXX XXXX

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

Bits 31:29 **CODE_OP[2:0]**: Flash memory operation code
- 000: No flash operation on going during previous reset
- 001: Single write operation interrupted
- 010: OBK alternate sector erase
- 011: Sector erase operation interrupted
- 100: Bank erase operation interrupted
- 101: Mass erase operation interrupted
- 110: Option change operation interrupted
- 111: OBK swap sector request

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **OTP_OP**: OTP operation interrupted
Indicates that reset interrupted an ongoing operation in OTP area (or OBKeys area).

Bit 23 **SYSF_OP**: Operation in system flash memory interrupted
Indicates that reset interrupted an ongoing operation in system flash.

Bit 22 **BK_OP**: Interrupted operation bank
It indicates which bank was concerned by operation.

Bit 21 **DATA_OP**: Flash high-cycle data area operation interrupted
It indicates if flash high-cycle data area is concerned by operation.

Bit 20 Reserved, must be kept at reset value.

Bits 19:0 **ADDR_OP[19:0]**: Interrupted operation address

7.11.8 FLASH option control register (FLASH_OPTCR)

This register is non-secure. It can be read and written by both secure and non-secure access, and protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait states when no memory operations are ongoing. The FLASH_OPTCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it while the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x01C

Reset value: 0xX000 0001

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

Bit 31 **SWAP_BANK**: Bank swapping option configuration bit
SWAP_BANK controls whether Bank1 and Bank2 are swapped or not. This bit is loaded with the SWAP_BANK bit of FLASH_OPTSR_CUR register only after reset or POR.
0: Bank1 and Bank2 not swapped
1: Bank1 and Bank2 swapped

Bits 30:2 Reserved, must be kept at reset value.

Bit 1 **OPTSTRT**: Option byte start change option configuration bit
OPTSTRT triggers an option byte change operation. The user can set OPTSTRT only when the OPTLOCK bit is cleared to 0. It is set only by Software and cleared when the option byte change is completed or an error occurs (PGSERR or OPTCHANGEERR). It is reset at the same time as BSY bit.
The user application cannot modify any FLASH_XXX_PRG flash interface register until the option change operation has been completed.
Before setting this bit, the user has to write the required values in the FLASH_XXX_PRG registers. The FLASH_XXX_PRG registers are locked until the option byte change operation has been executed in nonvolatile memory.

Bit 0 **OPTLOCK**: FLASH_OPTCR lock option configuration bit

The OPTLOCK bit locks the FLASH_OPTCR register as well as all _PRG registers. The correct write sequence to FLASH_OPTKEYR register unlocks this bit. If a wrong sequence is executed, or the unlock sequence to FLASH_OPTKEYR is performed twice, this bit remains locked until next system reset.

It is possible to set OPTLOCK by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When OPTLOCK changes from 0 to 1, the others bits of FLASH_OPTCR register do not change.

0: FLASH_OPTCR register unlocked

1: FLASH_OPTCR register locked.

7.11.9 FLASH non-secure status register (FLASH_NSSR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x020

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPT CHANGE ERR	OBKW ERR	OBK ERR	INC ERR	STRB ERR	PGS ERR	WRP ERR	EOP
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBNE	Res.	WBNE	BSY
												r		r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERR**: Option byte change error flag

OPTCHANGEERR flag indicates that an error occurred during an option byte change operation. When OPTCHANGEERR is set to 1, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTCHANGEERRIE bit of FLASH_NSCR register is set to 1.

Writing 1 to CLR_OPTCHANGEERR of register FLASH_NSCCR clears OPTCHANGEERR.

0: no option byte change errors occurred

1: one or more errors occurred during an option byte change operation.

Note: The OPTSTRT bit in FLASH_OPTCR cannot be set while OPTCHANGEERR is set.

Bit 22 **OBKWERR**: OBK write error flag

OBKWERR flag is raised when the address is not virgin on a write access to the OBK storage. Alternatively also when the OBK selector in the alternate sector is not virgin during a swap operation.

0: no OBK write error occurred

1: an OBK write error occurred

Bit 21 **OBKERR**: OBK general error flag

OBKERR flag is raised when the OBK-HDPL signal from the SBS does not match the HDPL value associated with the key slot during access to the key location. Alternatively also when the ALT_SECT is unexpectedly changed while the write buffer is being filled.

0: no OBK general error occurred

1: an OBK general error occurred

Bit 20 **INCERR:** inconsistency error flag

NSINCERR flag is raised when a inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1. Writing 1 to CLR_INCERR bit in the FLASH_NSCCR register clears NSINCERR.

0: no inconsistency error occurs

1: a inconsistency error occurs

Bit 19 **STRBERR:** strobe error flag

STRBERR flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set to 1. Writing 1 to CLR_STRBERR bit in FLASH_NSCCR register clears STRBERR.

0: no strobe error occurred

1: a strobe error occurred

Bit 18 **PGSERR:** programming sequence error flag

PGSERR flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Writing 1 to CLR_PGSERR bit in FLASH_NSCCR register clears PGSERR.

0: no sequence error occurred

1: a sequence error occurred

Bit 17 **WRPERR**: write protection error flag

WRPERR flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Writing 1 to CLR_WRPERR bit in FLASH_NSCCR register clears WRPERR.

0: no write protection error occurred

1: a write protection error occurred

Bit 16 **EOP**: end of operation flag

EOP flag is set when a operation (program/erase) completes. An interrupt is generated if the EOPIE is set to 1. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared by writing 1 to CLR_EOP bit in FLASH_NSCCR register.

0: no operation completed

1: a operation completed

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: data buffer not empty flag

DBNE flag is set when the flash interface is processing 6-bits ECC data in dedicated buffer. This bit cannot be set to 0 by software. The hardware resets it once the buffer is free.

0: data buffer not used

1: data buffer used, wait

Bit 2 Reserved, must be kept at reset value.

Bit 1 **WBNE**: write buffer not empty flag

WBNE flag is set when the flash interface is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the events below:

– the application software forces the write operation using FW bit in FLASH_NSCR
– the memory detects an error that involves data loss

This bit cannot be reset by software writing 0 directly. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer empty or full

1: write buffer waiting data to complete

Bit 0 **BSY**: busy flag

BSY flag indicates that a flash memory is busy by an operation (write, erase, option byte change, OBK operation). It is set at the beginning of a flash memory operation and cleared when the operation finishes, or an error occurs.

0: no programming, erase or option byte change operation being executed

1: programming, erase or option byte change operation being executed

7.11.10 FLASH secure status register (FLASH_SECSR)

This register is secure. It can be read only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x024

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBNE	Res.	WBNE	BSY
												r		r	r

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

Bit 22 **OBKWERR**: OBK write error flag

OBKWERR flag is raised when the address is not virgin on a write access to the OBK storage. Alternatively also when the OBK selector in the alternate sector is not virgin during a swap operation.

0: no OBK write error occurred

1: an OBK write error occurred

Bit 21 **OBKERR**: OBK general error flag

OBKERR flag is raised when the OBK-HDPL signal from the SBS does not match the HDPL value associated with the key slot during access to the key location. Alternatively also when the ALT_SECT is unexpectedly changed while the write buffer is being filled.

0: no OBK general error occurred

1: an OBK general error occurred

Bit 20 **INCERR:** inconsistency error flag

INCERR flag is raised when a inconsistency error occurs. An interrupt is generated if INCERRIE is set to 1. Writing 1 to CLR_INCERR bit in the FLASH_SECCCR register clears INCERR.

0: no inconsistency error occurred

1: a inconsistency error occurred

Bit 19 **STRBERR:** strobe error flag

STRBERR flag is raised when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set to 1. Writing 1 to CLR_STRBERR bit in FLASH_SECCCR register clears STRBERR.

0: no strobe error occurred

1: a strobe error occurred

Bit 18 **PGSERR:** programming sequence error flag

PGSERR flag is raised when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set to 1. Writing 1 to CLR_PGSERR bit in FLASH_SECCCR register clears PGSERR.

0: no sequence error occurred

1: a sequence error occurred

Bit 17 **WRPERR**: write protection error flag

WRPERR flag is raised when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set to 1. Writing 1 to CLR_WRPERR bit in FLASH_SECCCR register clears WRPERR.

0: no write protection error occurred

1: a write protection error occurred

Bit 16 **EOP**: end of operation flag

EOP flag is set when a operation (program/erase) completes. An interrupt is generated if the EOPIE is set to. It is not necessary to reset EOP before starting a new operation. EOP bit is cleared by writing 1 to CLR_EOP bit in FLASH_SECCCR register.

0: no operation completed

1: a operation completed

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **DBNE**: data buffer not empty flag

DBNE flag is set when the memory interface is processing 6-bits ECC data in dedicated buffer. This bit cannot be set to 0 by software. The hardware resets it once the buffer is free.

0: data buffer not used

1: data buffer used, wait

Bit 2 Reserved, must be kept at reset value.

Bit 1 **WBNE**: write buffer not empty flag

WBNE flag is set when the flash interface is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the events below:

- the application software forces the write operation using FW bit in FLASH_SECCR
- the flash interface detects an error that involves data loss

This bit cannot be reset by writing 0 directly by software. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.

0: write buffer empty or full

1: write buffer waiting data to complete

Bit 0 **BSY**: busy flag

BSY flag indicates that a FLASH memory is busy (write, erase, option byte change, OBK operations). It is set at the beginning of a flash memory operation and cleared when the operation finishes or an error occurs.

0: no programming, erase or option byte change operation being executed

1: programming, erase or option byte change operation being executed

7.11.11 FLASH non-secure control register (FLASH_NSCR)

This register is non-secure. It can be read and written by both secure and non-secure accesses, and can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait states when no memory operations are ongoing. The FLASH_NSCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x028

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPT CHANGE ERRIE	OBKW ERRIE	OBK ERRIE	INC ERRIE	STRB ERRIE	PGS ERRIE	WRP ERRIE	EOPIE
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER	Res.	Res.	SNB[6:0]							STRT	FW	BER	SER	PG	LOCK
rw			rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rs

Bit 31 **BKSEL**: Bank selector bit

Can be programmed only when LOCK is cleared to 0. The bit selects physical bank, SWAP_BANK setting is ignored.

0: Bank1 is selected for Bank erase / sector erase / interrupt enable

1: Bank2 is selected for BER / SER

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **OPTCHANGEERRIE**: Option byte change error interrupt enable bit

This bit controls if an interrupt must be generated when an error occurs during an option byte change. It can be programmed only when LOCK bit is cleared to 0.

0: no interrupt is generated when an error occurs during an option byte change

1: an interrupt is generated when and error occurs during an option byte change.

Bit 22 **OBKWERRIE**: OBK write error interrupt enable bit

OBKWERRIE enables generation of interrupt in case of OBK specific write error. This bit can be programmed only when LOCK bit is cleared to 0.

0: no interrupt is generated on OBK write error

1: an interrupt is generated on OBK write error

Bit 21 **OBKERRIE**: OBK general error interrupt enable bit

OBKERRIE enables generating an interrupt in case of OBK specific access error. This bit can be programmed only when LOCK bit is cleared to 0.

0: no interrupt is generated on OBK general access error

1: an interrupt is generated on OBK general access error

Bit 20 **INCERRIE**: inconsistency error interrupt enable bit

When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a inconsistency error occurs

1: interrupt generated when a inconsistency error occurs.

Bit 19 **STRBERRIE**: strobe error interrupt enable bit

When STRBERRIE bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a strobe error occurs

1: interrupt generated when strobe error occurs.

Bit 18 **PGSERRIE**: programming sequence error interrupt enable bit

When this bit is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a sequence error occurs

1: interrupt generated when sequence error occurs

Bit 17 **WRPERRIE**: write protection error interrupt enable bit

When this bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a protection error occurs

1: interrupt generated when a protection error occurs

Bit 16 **EOPIE**: end of operation interrupt control bit

Setting EOPIE bit to 1 enables the generation of an interrupt at the end of a program or erase operation. EOPIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated at the end of operation.

1: interrupt enabled when at the end of operation

Bit 15 **MER**: mass erase request

Setting MER bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0.

If BER or SER are both set, a PGSERR is raised.

0: mass erase not requested

1: mass erase requested

Note: An error is triggered when a mass erase is required and some sectors are protected.

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:6 **SNB[6:0]**: sector erase selection number

These bits are used to select the target sector for an erase operation, otherwise they are not used. They can be programmed only when LOCK is cleared to 0.

0x00: Sector 0 selected

0x01: Sector 1 selected

..

0x7F: Sector 127 selected

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

Bit 5 **STRT**: erase start control bit

STRT bit is used to start a sector erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.

STRT is reset at the end of the operation or when an error occurs. It cannot be reset by software.

Bit 4 **FW:** write forcing control bit

FW forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0.

The memory resets FW when the corresponding operation has been acknowledged.

Note: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.

Write forcing is effective only if the write buffer is not empty and was filled by non-secure access (in particular, FW does not start several write operations when the force-write operations are performed consecutively).

Since there is just one write buffer, FW can force a write in Bank1 or Bank2.

Bit 3 **BER**: erase request

Setting BER bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0.

If MER and SER are also set, a PGSERR is raised.

0: bank erase not requested

1: bank erase requested

Note: Write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2 **SER**: sector erase request

Setting SER bit to 1 requests a sector erase. SER can be programmed only when LOCK is cleared to 0.

If MER and SER are also set, a PGSERR is raised.

0: sector erase not requested

1: sector erase requested

Bit 1 **PG**: programming control bit

PG can be programmed only when LOCK is cleared to 0.

PG allows programming in Bank1 and Bank2.

0: programming disabled

1: programming enabled

Bit 0 **LOCK**: configuration lock bit

This bit locks the FLASH_NSCR register. The correct write sequence to FLASH_NSKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_NSKEYR is performed twice, this bit remains locked until the next system reset.

LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_NSCR register do not change.

0: FLASH_NSCR register unlocked

1: FLASH_NSCR register locked

7.11.12 FLASH secure control register (FLASH_SECCR)

This register is secure, can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Access: No wait states when no memory operations are ongoing. The FLASH_SECCR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x02C

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKSEL	Res.	INV	Res.	Res.	Res.	Res.	Res.	Res.	OBKW ERRIE	OBK ERRIE	INC ERRIE	STRB ERRIE	PGS ERRIE	WRP ERRIE	EOPIE
rw		rw							rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MER	Res.	Res.	SNB[6:0]							STRT	FW	BER	SER	PG	LOCK
rw			rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rs

Bit 31 **BKSEL**: Bank selector bit

BKSEL can only be programmed when LOCK is cleared to 0. The bit selects physical bank, SWAP_BANK setting is ignored.

0: Bank1 is selected for Bank erase / sector erase / interrupt enable

1: Bank2 is selected for BER / SER

Bit 30 Reserved, must be kept at reset value.

Bit 29 **INV**: Flash memory security state invert.

This bit inverts the flash memory security state.

Bits 28:23 Reserved, must be kept at reset value.

Bit 22 **OBKWERRIE**: OBK write error interrupt enable bit

OBKWERRIE enables generation of interrupt in case of OBK specific write error. OBKWERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt is generated on OBK write error

1: an interrupt is generated on OBK write error

Bit 21 **OBKERRIE**: OBK general error interrupt enable bit

OBKERRIE enables generating an interrupt in case of OBK specific access error. OBKERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt is generated on OBK general access error

1: an interrupt is generated on OBK general access error

Bit 20 **INCERRIE**: inconsistency error interrupt enable bit

When INCERRIE bit is set to 1, an interrupt is generated when an inconsistency error occurs during a write operation. INCERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a inconsistency error occurs

1: interrupt generated when a inconsistency error occurs.

Bit 19 **STRBERRIE**: strobe error interrupt enable bit

When STRBERRIE bit is set to 1, an interrupt is generated when a strobe error occurs (the master programs several times the same byte in the write buffer) during a write operation. STRBERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a strobe error occurs

1: interrupt generated when strobe error occurs.

Bit 18 **PGSERRIE**: programming sequence error interrupt enable bit

When PGSERRIE is set to 1, an interrupt is generated when a sequence error occurs during a program operation. PGSERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a sequence error occurs

1: interrupt generated when sequence error occurs

Bit 17 **WRPERRIE**: write protection error interrupt enable bit

When WRPERRIE bit is set to 1, an interrupt is generated when a protection error occurs during a program operation. WRPERRIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated when a protection error occurs

1: interrupt generated when a protection error occurs

Bit 16 **EOPIE**: end of operation interrupt control bit

Setting EOPIE bit to 1 enables the generation of an interrupt at the end of a program/erase operation. EOPIE can be programmed only when LOCK is cleared to 0.

0: no interrupt generated at the end of operation.

1: interrupt enabled when at the end of operation

Bit 15 **MER**: mass erase request

Setting MER bit to 1 requests a mass erase operation (user flash memory only). MER can be programmed only when LOCK is cleared to 0.

If BER or SER are also set, a PGSERR is raised.

0: mass erase not requested

1: mass erase requested

Note: An error is triggered when a mass erase is required and some sectors are protected.

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:6 **SNB[6:0]**: sector erase selection number

These bits are used to select the target sector for an erase operation, otherwise they are unused. SNB can be programmed only when LOCK is cleared to 0.

0x00: Sector 0 selected

0x01: Sector 1 selected

..

0x7F: Sector 127 selected

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

Bit 5 **STRT**: erase start control bit

STRT bit is used to start a sector erase or a bank erase operation. STRT can be programmed only when LOCK is cleared to 0.

STRT is reset at the end of the operation or when an error occurs. It cannot be reset by software.

Bit 4 **FW:** write forcing control bit

FW forces a write operation even if the write buffer is not full. In this case all bits not written are set to 1 by hardware. FW can be programmed only when LOCK is cleared to 0.

The memory resets FW when the corresponding operation has been acknowledged.

Note: Using a force-write operation prevents the application from updating later the missing bits with something else than 1, because it is likely that it leads to permanent ECC error.

Write forcing is effective only if the write buffer is not empty and was filled by secure access (in particular, FW does not start several write operations when the force-write operations are performed consecutively).

Since there is just one write buffer, FW can force a write in Bank1 or Bank2.

Bit 3 **BER**: erase request

Setting BER bit to 1 requests a bank erase operation (user flash memory only). BER can be programmed only when LOCK is cleared to 0.

If MER and SER are also set, a PGSERR is raised.

0: bank erase not requested

1: bank erase requested

Note: A write protection error is triggered when a bank erase is required and some sectors are protected.

Bit 2 **SER**: sector erase request

Setting SER bit to 1 requests a sector erase. SER can be programmed only when LOCK is cleared to 0.

If BER and MER are also set, a PGSERR is raised.

0: sector erase not requested

1: sector erase requested

Bit 1 **PG**: programming control bit

PG can be programmed only when LOCK is cleared to 0.

PG allows programming in Bank1 and Bank2.

0: programming disabled

1: programming enabled

Bit 0 **LOCK**: configuration lock bit

This bit locks the FLASH_SECCR register. The correct write sequence to FLASH_SECKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_NSKEYR is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_SECCR register do not change.

0: FLASH_SECCR register unlocked

1: FLASH_SECCR register locked

7.11.13 FLASH non-secure clear control register (FLASH_NSCCR)

This register is non-secure. It can be written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_OPTCHANGEERR	CLR_OBKWERR	CLR_OBKERR	CLR_INCERR	CLR_STRBERR	CLR_PGSERR	CLR_WRPERR	CLR_EOP
								w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CLR_OPTCHANGEERR:** Clear the flag corresponding flag in FLASH_NSSR by writing this bit.

Bit 22 **CLR_OBKWERR**: OBKWERR flag clear bit.
Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_NSSR register.

Bit 21 **CLR_OBKERR**: OBKERR flag clear bit.
Setting this bit to 1 resets to 0 OBKERR flag in FLASH_NSSR register.

Bit 20 **CLR_INCERR**: INCERR flag clear bit
Setting this bit to 1 resets to 0 INCERR flag in FLASH_NSSR register.

Bit 19 **CLR_STRBERR**: STRBERR flag clear bit
Setting this bit to 1 resets to 0 STRBERR flag in FLASH_NSSR register.

Bit 18 **CLR_PGSERR**: PGSERR flag clear bit
Setting this bit to 1 resets to 0 PGSERR flag in FLASH_NSSR register.

Bit 17 **CLR_WRPERR**: WRPERR flag clear bit
Setting this bit to 1 resets to 0 WRPERR flag in FLASH_NSSR register.

Bit 16 **CLR_EOP**: EOP flag clear bit
Setting this bit to 1 resets to 0 EOP flag in FLASH_NSSR register.

Bits 15:0 Reserved, must be kept at reset value.

7.11.14 FLASH secure clear control register (FLASH_SECCCR)

This register is secure. It can be written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_ OBKW ERR	CLR_ OBK ERR	CLR_ INC ERR	CLR_ STRB ERR	CLR_ PGS ERR	CLR_ WRP ERR	CLR_ EOP
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

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

Bit 22 **CLR_OBKWERR**: OBKWERR flag clear bit
Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_SECSR register.

Bit 21 **CLR_OBKERR**: OBKWERR flag clear bit
Setting this bit to 1 resets to 0 OBKWERR flag in FLASH_SECSR register.

Bit 20 **CLR_INCERR**: INCERR flag clear bit
Setting this bit to 1 resets to 0 INCERR flag in FLASH_SECSR register.

Bit 19 **CLR_STRBERR**: STRBERR flag clear bit
Setting this bit to 1 resets to 0 STRBERR flag in FLASH_SECSR register.

Bit 18 **CLR_PGSERR**: PGSERR flag clear bit
Setting this bit to 1 resets to 0 PGSERR flag in FLASH_SECSR register.

Bit 17 **CLR_WRPERR**: WRPERR flag clear bit
Setting this bit to 1 resets to 0 WRPERR flag in FLASH_SECSR register.

Bit 16 **CLR_EOP**: EOP flag clear bit
Setting this bit to 1 resets to 0 EOP flag in FLASH_SECSR register.

Bits 15:0 Reserved, must be kept at reset value.

7.11.15 FLASH privilege configuration register (FLASH_PRIVCFGR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register can be read by both privileged and unprivileged access. NSPRIV is a non-secure bit. SPRIV is a secure bit. It can be written only by privileged mode.

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

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

Bit 1 **NSPRIV**: privilege attribute for non secure registers
0: access to non secure registers is always granted
1: access to non secure registers is denied in case of unprivileged access.

Bit 0 **SPRIV**: privilege attribute for secure registers
0: access to secure registers is always granted
1: access to secure registers is denied in case of unprivileged access.

7.11.16 FLASH non-secure OBK configuration register (FLASH_NSOBKCFGR)

Register is only available when TZ_STATE = 0xC3. This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can only be accessed by privilege code. This register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x040

Reset value: 0x01FF 0000

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **SWAP_OFFSET[8:0]**: Key index (offset /16 bits) pointing for next swap.
0x00: means that no OBK is copied from current to alternate OBK sector during SWAP operation.
0x01 means that only the first OBK data (128 bits) are copied from current to alternate OBK sector
0x02 means that the two first OBK data are copied
…
0x1FF: means that all OBK data (511) are copied

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **ALT_SECT_ERASE**: alternate sector erase bit
0: do not touch OBK sector
1: erase the alternate OBK sector
When ALT_SECT bit is set, use this bit to generate an erase command for the OBK alternate sector. It is set only by Software and cleared when the OBK swap operation is completed or an error occurs (PGSERR). It is reset at the same time as BUSY bit.

Bit 2 **ALT_SECT**: alternate sector bit
0: current OBK sector is mapped to OBK address range for access
1: alternate OBK sector is mapped to OBK address range for access
This bit must not change while filling the write buffer, otherwise an error (OBKERR) is generated

Bit 1 **SWAP_SECT_REQ**: OBK swap sector request bit
0: no swap requested
1: launch the sector swap
When set, all the OBKs not updated in the alternate sector are copied from current sector to alternate one.
The SWAP_OFFSET must have a minimum value to launched the swap. The minimum value is 16 for OBK-HDPL = 1, 144 for OBK-HDPL = 2, and 192 for OBK-HDPL = 3.

Bit 0 **LOCK**: OBKCFGR lock option configuration bit
This bit locks the register, the correct write sequence unlocks it. If a wrong sequence is executed, or if the unlock sequence is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_NSCR register do not change.
0: FLASH_NSOBKCFGR register unlocked
1: FLASH_NSOBKCFGR register locked

7.11.17 FLASH secure OBK configuration register (FLASH_SECOBKCFGR)

This register can be accessed only in secure privileged mode. The FLASH_SECOBKCFGR register is not accessible in write mode when the BSY bit is set. Any attempt to write to it with the BSY bit set causes the AHB bus to stall until the BSY bit is cleared.

Address offset: 0x044

Reset value: 0x01FF 0000

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **SWAP_OFFSET[8:0]**: key index (offset /16 bits) pointing for next swap.

0x0: no OBK is copied from current to alternate OBK sector during SWAP operation.
0x1: only the first OBK data (128 bits) are copied from current to alternate OBK sector
0x2: the two first OBK data are copied
…
0x1FF: the 511 first OBK data are copied

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **ALT_SECT_ERASE**: alternate sector erase bit

0: do not touch OBK sector
1: erase the alternate OBK sector
When ALT_SECT bit is set, use this bit to generate an erase command for the OBK alternate sector. It is set only by Software and cleared when the OBK swap operation is completed or an error occurs (PGSERR). It is reset at the same time as the BUSY bit.

Bit 2 **ALT_SECT**: alternate sector bit

0: current OBK sector is mapped to OBK address range for access
1: alternate OBK sector is mapped to OBK address range for access
This bit must not change while filling the write buffer, otherwise an error is generated

Bit 1 **SWAP_SECT_REQ**: OBK swap sector request bit

0: no swap requested
1: launch the sector swap
When set, all the OBKs not updated in the alternate sector are copied from current sector to alternate one.
SWAP_OFFSET must have a minimum value to launch the swap. The minimum value is 16 for OBK-HDPL = 1, 144 for OBK-HDPL = 2, and 192 for OBK-HDPL = 3.

Bit 0 **LOCK**: OBKCFGR lock option configuration bit

This bit locks the FLASH_OBKCFGR register. The correct write sequence to FLASH_SECOBKKEYR register unlocks this bit. If a wrong sequence is executed, or if the unlock sequence to FLASH_SECOBKKEYR is performed twice, this bit remains locked until the next system reset. LOCK can be set by programming it to 1. When set to 1, a new unlock sequence is mandatory to unlock it. When LOCK changes from 0 to 1, the other bits of FLASH_NSCR register do not change.
0: FLASH_OBKCFGR register unlocked
1: FLASH_OBKCFGR register locked

7.11.18 FLASH HDP extension register (FLASH_HDPEXTR)

All bits in this register are non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

The register can be modified only in HDPL ≤ 2.

The values in this register cannot be decremented. Attempts to write a value lower than the current one are ignored.

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_EXT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **HDP2_EXT[6:0]**: HDP area extension in 8 Kbytes sectors in Bank2. Extension is added after the HDP2_END sector (included).

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **HDP1_EXT[6:0]**: HDP area extension in 8 Kbytes sectors in Bank1. Extension is added after the HDP1_END sector (included).

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.19 FLASH option status register (FLASH_OPTSR_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Default value: 0x0030 5CD8 (value in case of double ECC issue during OBL).

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x050

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	BOOT_UBE[7:0]								IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV
r		r	r	r	r	r	r	r	r	r	r			r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
r	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bit 31 **SWAP_BANK**: Bank swapping option status bit
SWAP_BANK reflects whether Bank1 and Bank2 are swapped or not.
SWAP_BANK is loaded to SWAP_BANK of FLASH_OPTCR after a reset.
0: Bank1 and Bank2 not swapped
1: Bank1 and Bank2 swapped

Bit 30 Reserved, must be kept at reset value.

Bits 29:22 **BOOT_UBE[7:0]:** Available only on cryptography enabled devices.
Unique boot entry control, selects either ST or OEM iRoT for secure boot.
0xC3: ST-iRoT (system flash) selected
0xB4: OEM-iRoT (user flash) selected. In Open PRODUCT_STATE this value selects bootloader. Defaut value.

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option status bit
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option status bit
When set the independent watchdog IWDG is in system Stop mode.
0: Independent watchdog frozen in system Stop mode
1: Independent watchdog keep running in system Stop mode.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **IO_VDDIO2_HSLV**: High-speed IO at low V_{DDIO2} voltage configuration bit.
This bit can be set only with V_{DDIO2} below 2.7 V.
0: High-speed IO at low V_{DDIO2} voltage feature disabled (V_{DDIO2} can exceed 2.7 V)
1: High-speed IO at low V_{DDIO2} voltage feature enabled (V_{DDIO2} remains below 2.7 V)

Bit 16 **IO_VDD_HSLV**: High-speed IO at low V_{DD} voltage configuration bit.
This bit can be set only with V_{DD} below 2.7 V.
0: High-speed IO at low V_{DD} voltage feature disabled (V_{DD} can exceed 2.7 V)
1: High-speed IO at low V_{DD} voltage feature enabled (V_{DD} remains below 2.7 V)

Bits 15:8 **PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). See *Section 7.6.11*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option status bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option status bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option status bit
0: WWDG watchdog is controlled by hardware
1: WWDG watchdog is controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option status bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

Bit 2 **BORH_EN**: Brownout high enable
0: disable
1: enable

Bits 1:0 **BOR_LEV[1:0]**: Brownout level option status bit
These bits reflects the power level that generates a system reset.
00 or 11: BOR Level 1, the threshold level is low (~ 2.1 V)
01: BOR Level 2, the threshold level is medium (~ 2.4 V)
10: BOR Level 3, the threshold level is high (~ 2.7 V)

7.11.20 FLASH option status register (FLASH_OPTSR_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x054

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWAP_BANK	Res.	BOOT_UBE[7:0]								IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 **SWAP_BANK**: Bank swapping option configuration bit
SWAP_BANK option bit is used to configure whether the Bank1 and Bank2 are swapped or not. This bit is loaded with the SWAP_BANK bit of FLASH_OPTSR_CUR register after a reset.
0: Bank1 and Bank2 not swapped
1: Bank1 and Bank2 swapped

Bit 30 Reserved, must be kept at reset value.

Bits 29:22 **BOOT_UBE[7:0]:** Available only on cryptography enabled devices.
Unique boot entry control, selects either ST or OEM iRoT for secure boot.
0xC3: ST-iRoT (system flash) selected
0xB4: OEM-iRoT (user flash) selected.
In Open PRODUCT_STATE this value selects bootloader. Defaut value.

Bit 21 **IWDG_STDBY**: IWDG Standby mode freeze option status bit
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode.

Bit 20 **IWDG_STOP**: IWDG Stop mode freeze option status bit
When set the independent watchdog IWDG is in system Stop mode.
0: Independent watchdog frozen in system Stop mode
1: Independent watchdog keep running in system Stop mode.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **IO_VDDIO2_HSLV**: High-speed IO at low V_{DDIO2} voltage configuration bit.
This bit can be set only with V_{DDIO2} below 2.7 V.
0: High-speed IO at low V_{DDIO2} voltage feature disabled (V_{DDIO2} can exceed 2.7 V)
1: High-speed IO at low V_{DDIO2} voltage feature enabled (V_{DDIO2} remains below 2.7 V)

Bit 16 **IO_VDD_HSLV**: High-speed IO at low VDD voltage configuration bit.
This bit can be set only with V_{DD} below 2.7 V.
0: High-speed IO at low V_{DD} voltage feature disabled (V_{DD} can exceed 2.7 V)
1: High-speed IO at low V_{DD} voltage feature enabled (V_{DD} remains below 2.7 V)

Bits 15:8 **PRODUCT_STATE[7:0]**: Life state code (based on Hamming 8,4). See *Section 7.6.11*.

Bit 7 **NRST_STDBY**: Core domain Standby entry reset option configuration bit
0: a reset is generated when entering Standby mode on core domain
1: no reset generated when entering Standby mode on core domain.

Bit 6 **NRST_STOP**: Core domain Stop entry reset option configuration bit
0: a reset is generated when entering Stop mode on core domain
1: no reset generated when entering Stop mode on core domain.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **WWDG_SW**: WWDG control mode option configuration bit
0: WWDG watchdog is controlled by hardware
1: WWDG watchdog is controlled by software

Bit 3 **IWDG_SW**: IWDG control mode option configuration bit
0: IWDG watchdog is controlled by hardware
1: IWDG watchdog is controlled by software

Bit 2 **BORH_EN**: Brownout high enable configuration bit
0: disabled
1: enabled

Bits 1:0 **BOR_LEV[1:0]**: Brownout level option configuration bit
These bits reflects the power level that generates a system reset.
00 or 11: BOR Level 1, the threshold level is low (~ 2.1 V)
01: BOR Level 2, the threshold level is medium (~ 2.4 V)
10: BOR Level 3, the threshold level is high (~ 2.7 V)

7.11.21 FLASH non-secure EPOCH register (FLASH_NSEPOCHR_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x060

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **NS_EPOCH[23:0]:** Non-volatile non-secure EPOCH counter

7.11.22 FLASH secure EPOCH register (FLASH_SECEPOCHR_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x068

Reset value: 0xXXXX XXXX

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **SEC_EPOCH[23:0]:** Non-volatile secure EPOCH counter

7.11.23 FLASH option status register 2 (FLASH_OPTSR2_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Default value: 0xB400 0170

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x070

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPD_DIS	Res.	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	SRAM2_RST	SRAM13_RST	Res.	Res.
							r		r	r	r	r	r		

Bits 31:24 **TZEN[7:0]**: TrustZone enable configuration bits
This bit enables the device is in TrustZone mode during an option byte change.
0xC3: TrustZone disabled
0xB4: TrustZone enabled.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **USBPD_DIS**: USB power delivery configuration option bit
0: Enabled
1: Disabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bit 5 **SRAM3_ECC**: SRAM3 ECC detection and correction disable
0: SRAM3 ECC check enabled
1: SRAM3 ECC check disabled

Bit 4 **BKPRAM_ECC**: Backup RAM ECC detection and correction disable
0: BKPRAM ECC check enabled
1: BKPRAM ECC check disabled

Bit 3 **SRAM2_RST**: SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs.

Bit 2 **SRAM13_RST**: SRAM1 and SRAM3 erase upon system reset
0: SRAM1 and SRAM3 erased when a system reset occurs
1: SRAM1 and SRAM3 not erased when a system reset occurs

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

7.11.24 FLASH option status register 2 (FLASH_OPTSR2_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits. Values after reset reflects the current values of the corresponding option bits.

Address offset: 0x074

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPD_DIS	Res.	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	SRAM2_RST	SRAM1_3_RST	Res.	Res.
							rw		rw	rw	rw	rw	rw		

Bits 31:24 **TZEN[7:0]**: TrustZone enable configuration bits
This bit enables the device is in TrustZone mode during an option byte change.
0xC3: TrustZone disabled
0xB4: TrustZone enabled

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **USBPD_DIS**: USB power delivery configuration option bit
0: Enabled
1: Disabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **SRAM2_ECC**: SRAM2 ECC detection and correction disable
0: SRAM2 ECC check enabled
1: SRAM2 ECC check disabled

Bit 5 **SRAM3_ECC**: SRAM3 ECC detection and correction disable
0: SRAM3 ECC check enabled
1: SRAM3 ECC check disabled

Bit 4 **BKPRAM_ECC**: Backup RAM ECC detection and correction disable
0: BKPRAM ECC check enabled
1: BKPRAM ECC check disabled

Bit 3 **SRAM2_RST**: SRAM2 erase when system reset
0: SRAM2 erased when a system reset occurs
1: SRAM2 not erased when a system reset occurs.
Note: SRAM erase is triggered by option byte change operation, when enabling this feature.

Bit 2 **SRAM1_3_RST**: SRAM1 and SRAM3 erase upon system reset
0: SRAM1 and SRAM3 erased when a system reset occurs
1: SRAM1 and SRAM3 not erased when a system reset occurs
Note: SRAM erase is triggered by option byte change operation, when enabling this feature.

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

7.11.25 FLASH non-secure boot register (FLASH_NSBOOTR_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of the corresponding option bits.

Address offset: 0x080

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

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

Bits 31:8 **NSBOOTADD[23:0]**: Non secure unique boot entry address
These bits reflect the Non secure UBE address

Bits 7:0 **NSBOOT_LOCK[7:0]:** Field locking the values of SWAP_BANK, and NSBOOTADD settings.
0xC3: SWAP_BANK and NSBOOTADD can still be modified following their individual rules.
0xB4: NSBOOTADD is frozen. SWAP_ BANK can be modified only with TZEN set to 0xB4 (enabled).

7.11.26 FLASH non-secure boot register (FLASH_NSBOOTR_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses, and protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x084

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

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

Bits 31:8 **NSBOOTADD[23:0]**: Non secure unique boot entry address
These bits allow configuring the Non secure BOOT address

Bits 7:0 **NSBOOT_LOCK[7:0]:** Field locking the values of SWAP_ BANK, and NSBOOTADD settings.
0xC3: SWAP_ BANK and NSBOOTADD can still be modified following their individual rules.
0xB4: NSBOOTADD is frozen. SWAP_ BANK can be modified only with TZEN set to 0xB4 (enabled).

7.11.27 FLASH secure boot register (FLASH_SECBOOTR_CUR)

This register is secure. It can be read only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of the corresponding option bits.

Address offset: 0x088

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

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

Bits 31:8 **SECBOOTADD[23:0]**: Unique boot entry secure address
These bits reflect the Secure UBE address

Bits 7:0 **SECBOOT_LOCK[7:0]:** Field locking the values of UBE, SWAP_BANK, and SECBOOTADD settings.
0xC3: BOOT_UBE, SWAP_BANK and SECBOOTADD can still be modified following their individual rules.
0xB4: BOOT_UBE and SECBOOTADD are frozen. SWAP_ BANK can be modified only with TZEN set to 0xC3 (disabled).

7.11.28 FLASH secure boot register (FLASH_BOOTR_PRG)

This register is secure, can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x08C

Reset value: 0xXXXX XXXX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

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

Bits 31:8 **SECBOOTADD[23:0]**: Secure unique boot entry address.
These bits allow configuring the secure UBE address.

Bits 7:0 **SECBOOT_LOCK[7:0]:** Field locking the values of UBE, SWAP_ BANK, and SECBOOTADD setting.
0xC3: BOOT_UBE, SWAP_ BANK and SECBOOTADD can still be modified following their individual rules.
0xB4: BOOT_UBE and SECBOOTADD are frozen. SWAP_BANK can be modified only with TZEN set to 0xC3 (disabled).

7.11.29 FLASH non-secure OTP block lock (FLASH_OTPBLR_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register reflects the current values of the corresponding option bits.

Address offset: 0x090

Reset value: 0xXXXX XXXX

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

Bits 31:0 **LOCKBL[31:0]**: OTP block lock

Block *n* corresponds to OTP 16-bit word 32 x *n* to 32 x *n* + 31.
LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP Block *n* are locked and attempt to program them results in WRPERR.
LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP Block *n* are not locked.
When one block is locked, it is not possible to remove the write protection.
Also if not locked, it is not possible to erase OTP words.

7.11.30 FLASH non-secure OTP block lock (FLASH_OTPBLR_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x094

Reset value: 0xXXXX XXXX

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

Bits 31:0 **LOCKBL[31:0]**: OTP block lock

Block *n* corresponds to OTP 16-bit word 32 x *n* to 32 x *n* + 31.
LOCKBL[n] = 1 indicates that all OTP 16-bit words in OTP Block *n* are locked and attempt to program them results in WRPERR.
LOCKBL[n] = 0 indicates that all OTP 16-bit words in OTP Block *n* are not locked.
When one block is locked, it is not possible to remove the write protection.
LOCKBL bits can be set if the corresponding bit in FLASH_OTPBLR_CUR is cleared.

7.11.31 FLASH secure block based register for Bank1 (FLASH_SECBB1Rx)

These registers (only one for STM32H523/33xx devices, as x = 1) are secure. They can be read and written only by secure access. A non-secure read/write access is RAZ/WI.

Address offset: 0x0A0 + 0x004 * (x-1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SECBB1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SECBB1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECBB1[31:0]**: Secure/non-secure 8 Kbytes flash Bank1 sector attributes (y = 0 to 31)

0: sector (32 * (x-1) + y) in Bank1 is non secure.
1: sector (32 * (x-1) + y) in Bank1 is secure.

7.11.32 FLASH privilege block based register for Bank1 (FLASH_PRIVBB1Rx)

These registers (only one for STM32H523/33xx devices, as x = 1) are privileged. They can be read and written only by a privileged access. This register can be protected against non-secure write access.

Address offset: 0x0C0 + 0x004 * (x-1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIVBB1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIVBB1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVBB1[31:0]**: Privileged/unprivileged 8-Kbyte flash Bank1 sector attribute (y = 0 to 31)

0: sectors (32 * (x-1) + y) in Bank1 is unprivileged.
1: sector (32 * (x-1) + y) in Bank1 is privileged.

7.11.33 FLASH security watermark for Bank1 (FLASH_SECWM1R_CUR)

This read-only register is non-secure, it can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register. It reflects the current values of the corresponding option bits.

Address offset: 0x0E0

Reset value: 0x00XX 00XX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **SECWM1_END[6:0]**: Bank1 security WM area 1 end sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **SECWM1_STRT[6:0]**: Bank1 security WM area 1 start sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.34 FLASH security watermark for Bank1 (FLASH_SECWM1R_PRG)

This register is secure, can be read and written only by secure access. A non-secure read/write access is RAZ/WI. This register can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register. This register is used to program values in corresponding option bits.

Address offset: 0x0E4

Reset value: 0x00XX 00XX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **SECWM1_END[6:0]**: Bank1 security WM area 1 end sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **SECWM1_STRT[6:0]**: Bank1 security WM area 1 start sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.35 FLASH write sector group protection for Bank1 (FLASH_WRP1R_CUR)

This register is non-secure. It can be read by both secure and non-secure accesses, and can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x0E8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG1[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRPSG1[31:0]**: Bank1 sector group protection option status byte

Each FLASH_WRP1R_CUR bit reflects the write protection status of the corresponding group of four consecutive sectors in Bank1 (0: the group is write-protected; 1: the group is not write-protected)

Bit 0: Group embedding sectors 0 to 3

Bit 1: Group embedding sectors 4 to 7

Bit N: Group embedding sectors 4 x N to 4 x N + 3

Bit 31: Group embedding sectors 124 to 127

Note: In STM32H523/33xx devices only bits [7:0] are used. Consider bits [31:8] as reserved, and keep them at reset value.

7.11.36 FLASH write sector group protection for Bank1 (FLASH_WRP1R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x0EC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **WRPSG1[31:0]**: Bank1 sector group protection option status byte

Setting WRPSG1 bits to 0 write protects the corresponding group of four consecutive sectors in Bank1 (0: the group is write-protected; 1: the group is not write-protected)

Bit 0: Group embedding sectors 0 to 3

Bit 1: Group embedding sectors 4 to 7

Bit N: Group embedding sectors 4 x N to 4 x N + 3

Bit 31: Group embedding sectors 124 to 127

Note: In STM32H523/33xx devices only bits [7:0] are used. Consider bits [31:8] as reserved, and keep them at reset value.

7.11.37 FLASH data sector configuration Bank1 (FLASH_EDATA1R_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. The register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of the corresponding options bits.

Address offset: 0x0F0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EDATA1_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1_STRT[2:0]		
r													r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EDATA1_EN:** Bank1 flash high-cycle data enable

0: No flash high-cycle data area

1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **EDATA1_STRT[2:0]:**

EDATA1_STRT contains the start sectors of the flash high-cycle data area in Bank1 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.

000: The last sector of Bank1 is reserved for flash high-cycle data

001: The two last sectors of Bank1 are reserved for flash high-cycle data

010: The three last sectors of Bank1 are reserved for flash high-cycle data

...

111: The eight last sectors of Bank1 are reserved for flash high-cycle data

7.11.38 FLASH data sector configuration Bank1 (FLASH_EDATA1R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding options bits.

Address offset: 0xF4

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EDATA1_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1_STRT[2:0]		
rw													rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EDATA1_EN:** Bank1 flash high-cycle data enable
0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **EDATA1_STRT[2:0]:**
EDATA1_STRT contains the start sectors of the flash high-cycle data area in Bank1 There is no hardware effect to those bits. They must be managed by ST tools in Flasher.
000: The last sector of Bank1 is reserved for flash high-cycle data
001: The two last sectors of Bank1 are reserved for flash high-cycle data
010: The three last sectors of Bank1 are reserved for flash high-cycle data
...
111: The eight last sectors of Bank1 are reserved for flash high-cycle data

7.11.39 FLASH HDP Bank1 configuration (FLASH_HDP1R_CUR)

This register is non-secure. It can be read by both secure and non-secure accesses, and protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register. This register is initially loaded with the values of the corresponding option bits.

Address offset: 0x0F8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **HDP1_END[6:0]:** HDPL barrier end set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **HDP1_STRT[6:0]:** HDPL barrier start set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.40 FLASH HDP Bank1 configuration (FLASH_HDP1R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure access, and can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

The register is accessible only in HDPL1. In HDPL2 or HDPL3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

Address offset: 0x0FC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **HDP1_END[6:0]:** HDPL barrier end set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **HDP1_STRT[6:0]:** HDPL barrier start set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.41 FLASH ECC correction register (FLASH_ECCCORR)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ECCC	Res.	Res.	Res.	Res.	ECCCIE	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Res.	Res.	Res.	Res.
	rw					rw	r	r	r	r	r				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR_ECC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ECCC:** ECC correction set by hardware when single ECC error has been detected and corrected.

Cleared by writing 1.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **ECCCIE**: ECC single correction error interrupt enable bit

When ECCCIE bit is set to 1, an interrupt is generated when an ECC single correction error occurs during a read operation.

0: no interrupt generated when an ECC single correction error occurs

1: non-secure interrupt generated when an ECC single correction error occurs

Bit 24 **OTP_ECC**: OTP ECC error bit

This bit is set to 1 when one single ECC correction occurred during the last successful read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bitfield.

Bit 23 **SYSF_ECC**: ECC fail for corrected ECC error in system flash memory

It indicates if system flash memory is concerned by ECC error.

Bit 22 **BK_ECC**: ECC fail bank for corrected ECC error

It indicates which bank is concerned by ECC error

Bit 21 **EDATA_ECC**: ECC fail for corrected ECC error in flash high-cycle data area

It indicates if flash high-cycle data area is concerned by ECC error.

Bit 20 **OBK_ECC**: Single ECC error corrected in flash OB Keys storage area. It indicates the OBK storage concerned by ECC error.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address

When an ECC error occurs (for single correction) during a read operation, ADDR_ECC contains the address that generated the error. It is reset when the flag error is reset.

The flash interface programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an ECC error is saved.

The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

7.11.42 FLASH ECC detection register (FLASH_ECCDETR)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x104

Reset value: 0x0000 0000

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

Bit 31 **ECCD:** ECC detection
Set by hardware when two ECC error has been detected.
When this bit is set, a NMI is generated.
Cleared by writing 1. Needs to be cleared in order to detect subsequent double ECC errors.

Bits 30:25 Reserved, must be kept at reset value.

Bit 24 **OTP_ECC**: OTP ECC error bit
This bit is set to 1 when double ECC detection occurred during the last read operation from the read-only/ OTP area. The address of the ECC error is available in ADDR_ECC bitfield.

Bit 23 **SYSF_ECC**: ECC fail for double ECC error in system flash memory
It indicates if system flash memory is concerned by ECC error.

Bit 22 **BK_ECC**: ECC fail bank for double ECC error
It indicates which bank is concerned by ECC error

Bit 21 **EDATA_ECC**: ECC fail double ECC error in flash high-cycle data area
It indicates if flash high-cycle data area is concerned by ECC error.

Bit 20 **OBK_ECC**: ECC fail double ECC error in flash OB Keys storage area. It indicates the OBK storage concerned by ECC error.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDR_ECC[15:0]**: ECC error address
When an ECC error occurs (double detection) during a read operation, the ADDR_ECC contains the address that generated the error. It is reset when the flag error is reset.
The flash interface programs the address in this register only when no ECC error flags are set. This means that only the first address that generated an double ECC error is saved.
The address in ADDR_ECC is relative to the flash memory area where the error occurred (user flash memory, system flash memory, data area, read-only/OTP area).

7.11.43 FLASH ECC data (FLASH_ECCDR)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x108

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DATA_ECC[15:0]**: ECC error data

When an double detection ECC error occurs on special areas with 6-bit ECC on 16-bit data (data, read-only/OTP), the failing data is read to this register.

By checking if it is possible to determine whether the failure was on a real data, or due to access to uninitialized memory.

7.11.44 FLASH secure block-based register for Bank2 (FLASH_SECBB2Rx)

These registers (only one for STM32H523/33xx devices, as x = 1) are secure, they can be read and written only by secure access. A non-secure read/write access is RAZ/WI. The registers can be protected against unprivileged access when SPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1A0 + 0x004 * (x-1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SECBB2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SECBB2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECBB2[31:0]**: Secure/non-secure flash Bank2 sector attribute (y = 0 to 31)

0: sectors (32 * (x-1) + y) in Bank2 is non secure

1: sector (32 * (x-1) + y) in Bank2 is secure

7.11.45 FLASH privilege block-based register for Bank2 (FLASH_PRIVBB2Rx)

These registers (only one for STM32H523/33xx devices, as x = 1) are privileged, they can be read and written only by a privileged access, and can be protected against non-secure write access by watermark.

Address offset: 0x1C0 + 0x004 * (x-1), (x = 1 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIVBB2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIVBB2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVBB2[31:0]:** Privileged / non-privileged 8-Kbyte flash Bank2 sector attribute (y = 0 to 31)
0: sectors (32 * (x-1) + y) in Bank2 is unprivileged
1: sector (32 * (x-1) + y) in Bank2 is privileged

7.11.46 FLASH security watermark for Bank2 (FLASH_SECWM2R_CUR)

This register is non-secure, it can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x1E0

Reset value: 0x00XX 00XX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **SECWM2_END[6:0]**: Bank2 security WM end sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **SECWM2_STRT[6:0]**: Bank2 security WM area start sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.47 FLASH security watermark for Bank2 (FLASH_SECWM2R_PRG)

This register is secure, can be read and written only by secure access. A non-secure read/write access is RAZ/WI. The register can be protected against unprivileged accesses when SPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x1E4

Reset value: 0x00XX 00XX

Bits 0 to 31 are loaded with values from the flash memory at OBL.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **SECWM2_END[6:0]**: Bank2 security WM area end sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **SECWM2_STRT[6:0]**: Bank2 security WM area start sector

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.48 FLASH write sector group protection for Bank2 (FLASH_WRP2R_CUR)

This register is non-secure, can be read by both secure and non-secure access, and protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of the corresponding option bits.

Address offset: 0x1E8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG2[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRPSG2[31:0]**: Bank2 sector group protection option status byte

Each bit reflects the write protection status of the corresponding group of four consecutive sectors in Bank2 (0: group is write-protected; 1: group is not write-protected)

Bit 0: Group embedding sectors 0 to 3

Bit 1: Group embedding sectors 4 to 7

Bit N: Group embedding sectors 4 x N to 4 x N + 3

Bit 31: Group embedding sectors 124 to 127

Note: In STM32H523/33xx devices only bits [7:0] are used. Consider bits [31:8] as reserved, and keep them at reset value.

7.11.49 FLASH write sector group protection for Bank2 (FLASH_WRP2R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This register is used to program values in corresponding option bits.

Address offset: 0x1EC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPSG2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPSG2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **WRPSG2[31:0]**: Bank2 sector group protection option status byte
Setting WRPSGn2 bits to 0 write protects the corresponding group of four consecutive sectors in Bank2 (0: group is write-protected; 1: group is not write-protected)
Bit 0: Group embedding sectors 0 to 3
Bit 1: Group embedding sectors 4 to 7
Bit N: Group embedding sectors 4 x N to 4 x N + 3
Bit 31: Group embedding sectors 124 to 127

Note: In STM32H523/33xx devices only bits [7:0] are used. Consider bits [31:8] as reserved, and keep them at reset value.

7.11.50 FLASH data sectors configuration Bank2 (FLASH_EDATA2R_CUR)

This register is non-secure, can be read by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

This read-only register reflects the current values of the corresponding options bits.

Address offset: 0x1F0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EDATA2_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2_STRT[2:0]		
r													r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EDATA2_EN:** Bank2 flash high-cycle data enable
0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **EDATA2_STRT[2:0]:**
EDATA2_STRT contains the start sectors of the flash high-cycle data area in Bank2 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
000: The last sector of Bank2 is reserved for flash high-cycle data
001: The two last sectors of Bank2 are reserved for flash high-cycle data
010: The three last sectors of Bank2 are reserved for flash high-cycle data
...
111: The eight last sectors of Bank2 are reserved for flash high-cycle data

7.11.51 FLASH data sector configuration Bank2 (FLASH_EDATA2R_PRG)

This register is non-secure. It can be read and written by both secure and non-secure accesses, and protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register. It is used to program values in corresponding options bits.

Address offset: 0x1F4

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EDATA2_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2_STRT[2:0]		
rw													rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EDATA2_EN:** Bank2 flash high-cycle data enable
0: No flash high-cycle data area
1: Flash high-cycle data is used

Bits 14:3 Reserved, must be kept at reset value.

Bits 2:0 **EDATA2_STRT[2:0]:**
EDATA2_STRT contains the start sectors of the flash high-cycle data area in Bank2 There is no hardware effect to those bits. They shall be managed by ST tools in Flasher.
000: The last sector of Bank2 is reserved for flash high-cycle data.
001: The two last sectors of Bank2 are reserved for flash high-cycle data.
010: The three last sectors of Bank2 are reserved for flash high-cycle data.
...
111: The eight last sectors of Bank2 are reserved for flash high-cycle data.

7.11.52 FLASH HDP Bank2 configuration (FLASH_HDP2R_CUR)

This register is initially loaded with the values of the corresponding option bits. This register is non-secure. It can be read by both secure and non-secure accesses, and can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Address offset: 0x1F8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[6:0]						
									r	r	r	r	r	r	r

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

Bits 22:16 **HDP2_END[6:0]:** HDPL barrier end set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **HDP2_STRT[6:0]:** HDPL barrier start set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.11.53 FLASH HDP Bank2 configuration (FLASH_HDP2R_PRG)

This register is non-secure. It can be read and written only by both secure and non-secure accesses. This register can be protected against unprivileged access when NSPRIV = 1 in the FLASH_PRIVCFGR register.

Register is accessible only in HDPL1. In HDPL2 or HDPL3 it is WI, RAZ.

This register is used to program values in corresponding option bits.

Address offset: 0x1FC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[6:0]						
									rw	rw	rw	rw	rw	rw	rw

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

Bits 22:16 **HDP2_END[6:0]:** HDPL barrier end set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

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

Bits 6:0 **HDP2_STRT[6:0]:** HDPL barrier start set in number of 8-Kbyte sectors

Note: For STM32H523/33xx devices the maximum value is 0x1F, consider bits [6:5] as reserved, and keep them at reset value.

7.12 FLASH register map and reset values

Table 85. Register map and reset value table

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	FLASH_ACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRFTEN	Res.	Res.	WRHIGHFREQ [1:0]		LATENCY [3:0]			
	Reset value																								0			0	1	0	0	1	1
0x004	FLASH_NSKEYR	NSKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x008	FLASH_ SECKEYR	SECKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	FLASH_OPTKEYR	OPTKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	FLASH_ NSOBKKEYR	NSOBKKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	FLASH_ SECOBKKEYR	SECOBKKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	FLASH_OPSR	CODE_OP [2:0]			Res.	Res.	Res.	Res.	OTP_OP	SYSF_OP	BK_OP	DATA_OP	Res.	ADDR_OP[19:0]																			
	Reset value	X	X	X					X	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x01C	FLASH_OPTCR	SWAP_BANK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTSTRT	OPTLOCK
	Reset value	0																														0	1
0x020	FLASH_NSSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGERR	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDBNE	Res.	WBNE	BSY
	Reset value									0	0	0	0	0	0	0	0													0		0	0
0x024	FLASH_SECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWERR	OBKERR	INCERR	STRBERR	PGSERR	WRPERR	EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBNE	Res.	WBNE	BSY
	Reset value										0	0	0	0	0	0	0													X		X	X
0x028	FLASH_NSCR	BKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPTCHANGEERRIE	OBKWIE	OBKIE	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	MER	Res.	Res.	SNB[6:0]							STRT	FW	BER	SER	PG	LOCK
	Reset value	0									0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	1

Table 85. Register map and reset value table (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	FLASH_SECCR	BKSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OBKWIE	OBKERRIE	INCERRIE	STRBERRIE	PGSERRIE	WRPERRIE	EOPIE	MER	Res.	Res.	SNB[6:0]							STRT	FW	BER	SER	PG	LOCK
	Reset value	0	0								0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	1
0x030	FLASH_NSCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_OPTCHANGEERR	CLR_OBKWERR	CLR_OBKERR	CLR_INCERR	CLR_STRBERR	CLR_PGSERR	CLR_WRPERR	CLR_EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0																
0x034	FLASH_ SECCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR_OBKWERR	CLR_OBKERR	CLR_INCERR	CLR_STRBERR	CLR_PGSERR	CLR_WRPERR	CLR_EOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	0	0	0	0	0	0																
0x03C	FLASH_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV
	Reset value																															0	0
0x040	FLASH_ NSOBKCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_OFFSET[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALT_SECT_ERASE	ALT_SECT	SWAP_SECT_REQ	LOCK
	Reset value								1	1	1	1	1	1	1	1	1													0	0	0	0
0x044	FLASH_ SECOBKCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP_OFFSET[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALT_SECT_ERASE	ALT_SECT	SWAP_SECT_REQ	LOCK
	Reset value								1	1	1	1	1	1	1	1	1													0	0	0	0
0x048	FLASH_HDPEXTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_EXT[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_EXT[6:0]						
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	FLASH_ OPTSR_CUR	SWAP_BANK	Res.	BOOT_UBE[7:0]								IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV	PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
	Reset value	X		X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X

Table 85. Register map and reset value table (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	FLASH_ OPTSR_PRG	SWAP_BANK	Res.	BOOT_UBE[7:0]								IWDG_STDBY	IWDG_STOP	Res.	Res.	IO_VDDIO2_HSLV	IO_VDD_HSLV	PRODUCT_STATE[7:0]								NRST_STDBY	NRST_STOP	Res.	WWDG_SW	IWDG_SW	BORH_EN	BOR_LEV[1:0]	
	Reset value	X		X	X	X	X	X	X	X	X	X	X			X	X	X	X	X	X	X	X	X	X	X	X		X	X	X	X	X
0x058	Reserved	Reserved																															
0x05C	Reserved	Reserved																															
0x060	FLASH_ NSEPOCHR_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NS_EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x064	FLASH_ NSEPOCHR_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NS_EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x068	FLASH_ SECEPOCHR_CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x06C	FLASH_ SECEPOCHR_PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC_EPOCH[23:0]																							
	Reset value									X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x070	FLASH_ OPTSR2_CUR	TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPD_DIS	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	SRAM2_RST	SRAM13_RST	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X																	X	X	X	X	X	X		
0x074	FLASH_ OPTSR2_PRG	TZEN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBPD_DIS	SRAM2_ECC	SRAM3_ECC	BKPRAM_ECC	SRAM2_RST	SRAM13_RST	Res.	Res.
	Reset value	X	X	X	X	X	X	X	X																	X	X	X	X	X	X		
0x080	FLASH_ NSBOOTR_CUR	NSBOOTADD[23:0]																								NSBOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x084	FLASH_ NSBOOTR_PRG	NSBOOTADD[23:0]																								NSBOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x088	FLASH_ SECBOOTR_CUR	SECBOOTADD[23:0]																								SECBOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08C	FLASH_ SECBOOTR_PRG	SECBOOTADD[23:0]																								SECBOOT_LOCK[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x090	FLASH_ OTPBLR_CUR	LOCKBL[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 85. Register map and reset value table (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x094	FLASH_ OTPBLR_PRG	LOCKBL[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0A0 + 0x004 * x (x = 1 to 4)	FLASH_ SECBB1R_x	SECBB1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0 + 0x004 * x (x = 1 to 4)	FLASH_ PRIVBB1R_x	PRIVBB1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D0 - 0x0DC	Reserved	Reserved																															
0x0E0	FLASH_SECWM1R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x0E4	FLASH_SECWM1R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_END1[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM1_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x0E8	FLASH_WRP1R _CUR	WRPSG1[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0EC	FLASH_WRP1R _PRG	WRPSG1[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0F0	FLASH_EDATA1R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1_STRT [2:0]		
	Reset value																	X													X	X	X
0x0F4	FLASH_EDATA1R _PRG	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA1_STRT [2:0]		
	Reset value																	X													X	X	X
0x0F8	FLASH_HDP1R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[6:0]						
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	1
0x0FC	FLASH_HDP1R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP1_STRT[6:0]						
	Reset value										X	X	X	X	X	X	0										X	X	X	X	X	X	X
0x100	FLASH_ECCCORR	Res.	ECCC	Res.	Res.	Res.	Res.	ECCIE	OTP_ECC	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	0x0000 0000		0					0	0	0	0		0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 85. Register map and reset value table (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x104	FLASH_ECCDETR	ECCD	Res.	Res.	Res.	Res.	Res.	Res.	OTP_ECC_FAIL	SYSF_ECC	BK_ECC	EDATA_ECC	OBK_ECC	Res.	Res.	Res.	Res.	ADDR_ECC[15:0]															
	Reset value	0							0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	FLASH_ECCDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA_ECC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A0 + 0x004 * x (x = 1 to 4)	FLASH_ SECBB2R_x	SECBB2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A0 + 0x004 * x (x = 1 to 4)	FLASH_ PRIVBB2R_x	PRIVBB2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1D0 - 0x1DC	Reserved	Reserved																															
0x1E0	FLASH_SECWM2R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x1E4	FLASH_SECWM2R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_END1[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SECWM2_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x1E8	FLASH_WRP2R _CUR	WRPSG2[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x1EC	FLASH_WRP2R _PRG	WRPSG2[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x1F0	FLASH_EDATA2R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2_STRT [2:0]		
	Reset value																	X													X	X	X
0x1F4	FLASH_EDATA2R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDATA2_STRT [2:0]		
	Reset value																	X													X	X	X
0x1F8	FLASH_HDP2R _CUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X
0x1FC	FLASH_HDP2R _PRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_END[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDP2_STRT[6:0]						
	Reset value										X	X	X	X	X	X	X										X	X	X	X	X	X	X

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

8 Instruction cache (ICACHE)

8.1 ICACHE introduction

The instruction cache (ICACHE) is introduced on the C-AHB code bus of the Cortex®-M33 processor, to improve performance when fetching instructions and data from internal and external memories.

Some specific features, like dual master ports, hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

8.2 ICACHE main features

The main features of ICACHE are listed below:

- Bus interface
  - One 32-bit AHB slave port, the execution port (input from Cortex®-M33 C-AHB code interface)
  - Two AHB master ports: master1 and master2 ports, with outputs to Fast and Slow buses of main AHB bus matrix, respectively
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Dual master access: feature used to decouple the traffic according to targeted memory. For example, the ICACHE assigns fast traffic (addressing flash memories and SRAMs) to the AHB master1 port, and slow traffic (addressing external memories) to the AHB master2 port, thus preventing processor stalls on line refills from external memories. This allows ISR (interrupt service routine) fetching on internal flash memory to take place in parallel with a cache line refill from external memories.
  - Minimal impact on interrupt latency, thanks to dual master
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, **w**, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Memory address remap
  - Possibility to remap input address falling into up to four memory regions (used to remap aliased code in external memories to the code region, for execution from the C-AHB code interface)
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls

  - Possibility to configure burst type of AHB memory transaction for remapped regions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error, and optionally, to raise an interrupt
- TrustZone® security support
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

8.3 ICACHE implementation

Table 86. ICACHE features

Feature	ICACHE
Number of ways	2
Cache size	8 Kbytes
Cache line width	16 bytes
Range granularity of memory regions to be remapped	2 Mbytes
Number of regions to remap	4
Data size of AHB slave interface	32 bits
Data size of AHB fast master1 interface	128 bits
Data size of AHB slow master2 interface	32 bits

8.4 ICACHE functional description

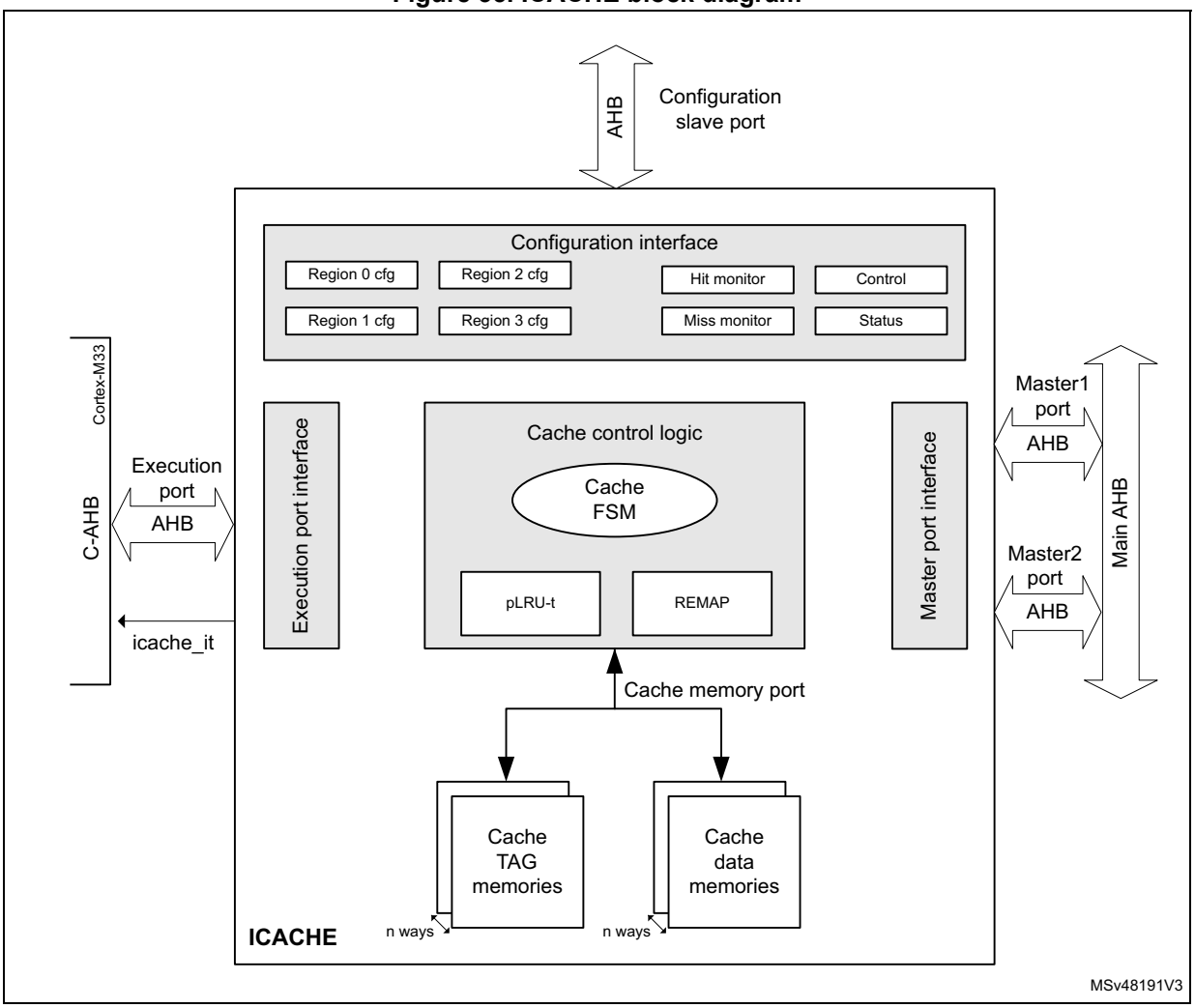
The purpose of this block is to cache instruction fetches or instruction memories loads, coming from the processor. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

8.4.1 ICACHE block diagram

Figure 35. ICACHE block diagram



8.4.2 ICACHE reset and clocks

The ICACHE is clocked on the Cortex®-M33 C-AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed, except the remapping mechanism that is still functional: slave input requests (remapped or not) are forwarded to the master port(s).

8.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:

- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 87* for B and W definitions):

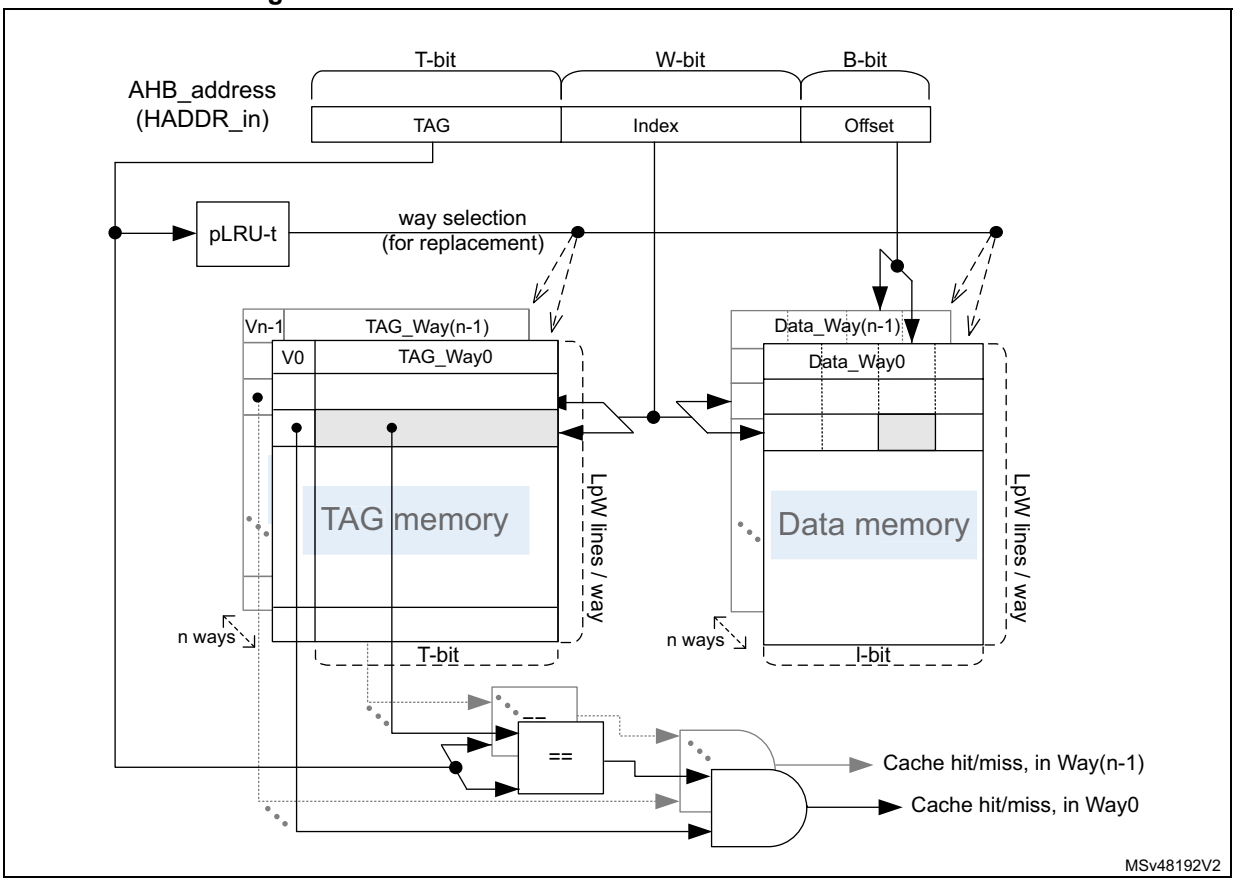
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 36* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 87. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 36. ICACHE TAG and data memories functional view



8.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 88. TAG memory dimensioning parameters for direct-mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, remapping, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 8.4.8*): only one way (the unique one) is possible for any data refill.

8.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master ports in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The address remapping is performed, whether the ICACHE is enabled or not, if the input transaction address falls into the memory regions defined and enabled in ICACHE_CRRx (see *Figure 37*).

The ICACHE is, by default, disabled at boot.

8.4.6 Cacheable and noncacheable traffic

The ICACHE is developed for the Cortex®-M33 core. It is placed on the C-AHB bus, and thus caches the code memory region, ranging from address 0x0000 0000 to 0x1FFF FFFF of the memory map.

To make other memory regions cacheable, the ICACHE supports a memory region remapping feature, used to define up to four external memory regions, whose addresses have an alias in the code region. Addressing these memory regions through their code alias address allows the memory request to be routed to the C-AHB bus, and to be managed by the ICACHE.

Any external memory space physically mapped at an address in the range [0x6000 0000:0xAFFF FFFF] or [0xC000 0000:0xDFFF FFFF] can be aliased with an address in the range [0x0000 0000:0x07FF FFFF] or [0x1000 0000:0x1FFF FFFF].

For a given memory request in the code region, the ICACHE implements the address remapping functionality first. If aliased, it is the remapped address, which is then cached, and, if needed, provided to the master port to address the main AHB bus matrix.
The destination physical address does not need further manipulation on the AHB bus.

The remapping functionality is available also for noncacheable traffic, and when the cache is disabled.

Additional details on address remapping are provided in *Section 8.4.7*.

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 89*. This AHB attribute depends on the MPU (memory protection unit) programming for the addressed region.

Table 89. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. The AHB transaction is propagated unchanged to the master output port, except the transaction address, which may be modified due to the address remapping feature (see *Section 8.4.7*).

The bypass, and eventual remap logic, does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in *Section 8.4.8*.

Cacheable memory regions are defined and programmed by the user in the MPU responsible for the generation of the AHB attribute signals for any transaction addressing a given region.

Table 90 summarizes programmable configurations of various memories.

Table 90. Memory configurations

Memory	Cacheable (MPU programming)	Remapped in the ICACHE (ICACHE_CRRx programming)
Flash	Yes or no	Not required
SRAM	Not recommended	Not required
External memories	Yes	Required
	No	Required if the user wants code in external memories fetched on C-AHB bus (else on S-AHB bus)

8.4.7 Address remapping

The ICACHE allows an alias address to be defined in the code region for up to four external memory regions.

The address remapping is applied on the code alias address, transforming it into the destination external physical address.

The remapping operation is fully software configurable by programming ICACHE_CRRx (x = 0 to 3, number of remapped regions). This programming can be done only when the ICACHE is disabled.

Each region x can be individually enabled with REN in ICACHE_CRRx. Once enabled, the remap operation occurs even if the ICACHE is disabled, or if the transaction is noncacheable.

Remap regions can have different size: each can be programmed in RSIZE of its ICACHE_CRRx. The size of each region is a power of two multiple of range granularity (2 Mbytes), with a minimum of 2 Mbytes, and a maximum of 128 Mbytes.

The address remapping mechanism is based on the matching of an incoming AHB address (HADDR_in) with a given code subregion base-address, and the modification of this address into its (remapped) physical address, as follows:

- HADDR_in belongs to region x if HADDR_in[31:RI] = 000:BASEADDR[28:RI], where:
  - 000:BASEADDR is the code subregion base-address programmed in BASEADDR of ICACHE_CRRx.
  - RI defines the number of significant bits to consider. RI = $\log_2$ (region size) with a minimum value of 21 (for a 2-Mbyte region) and a maximum value of 27 (for a 128-Mbyte region).
- If the region x is enabled, the master port output AHB address (HADDR_out) is composed by concatenating the two below parts:
  - REMAPADDR[31:RI] field of ICACHE_CRRx as MSBs
  - HADDR_in[RI-1:0] as LSBs

Figure 37 describes the matching and the output address generation.

Figure 37. ICACHE remapping address mechanism

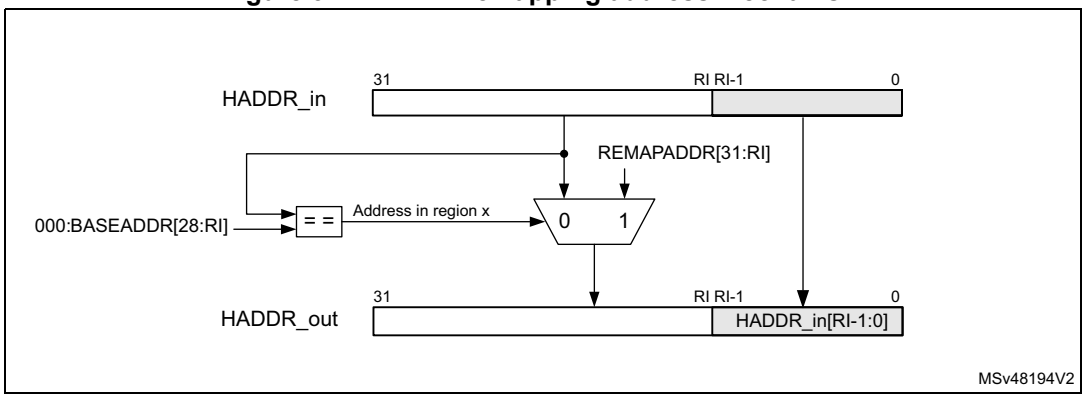


Table 91 summarizes all possible configurations of BASEADDR and REMAPADDR sizes (number of significant MSBs) in ICACHE_CRRx, depending upon RSIZE.

Table 91. ICACHE remap region size, base address, and remap address

Region size (Mbytes)	Base address size (MSBs)	Remap address (MSBs)
2	8	11
4	7	10
8	6	9
16	5	8
32	4	7
64	3	6
128	2	5

Care must be taken while programming BASEADDR and REMAPADDR in ICACHE_CRRx: if the programmed value is bigger than expected (number of MSBs, see *Table 91*), the unneeded extra LSBs are ignored.

Typical remapping example: a 128-Mbyte external RAM region physically located in the external address range [0x9000 0000:0x97FF FFFF], remapped in the code section range [0x1800 0000:0x1FFF FFFF]:

- REMAPADDR[31:21] = 0x480
- BASEADDR[28:21] = 0xC0
- HADDR_in[31:27] is compared to 000:BASEADDR[28:27], while HADDR_in and BASEADDR[26:21] are ignored.

If the comparison matches:

- HADDR_out[31:27] gets REMAPADDR[31:27] (in place of HADDR_in[31:27])
- HADDR_out[26:0] gets HADDR_in[26:0]

The software can program the kind of AHB burst generated by the ICACHE master ports on the bus matrix (for cache line refill), by setting HBURST in ICACHE_CRRx with:

- WRAP for remapped external memories accessed through interfaces that can support WRAP burst mode, providing the benefit of the critical-word-first feature performance:
  - WRAP burst size = cache line size
  - WRAP burst start address = word address of the first data requested by the core
- INCR: INCR burst mode for external memories accessed through the interfaces that do not support WRAP burst mode (losing the benefit of critical-word-first feature):
  - INCR burst size = cache line size
  - INCR burst start address = address aligned on the boundary of the cache line containing the requested word.

Note: Coherency is needed when programming the SAU (secure attribution unit) and the MPU (memory protection unit) attributes for both the external regions and their aliased code subregions.

8.4.8 Cacheable accesses

When the ICACHE receives a cacheable transaction from the Cortex®-M33, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words) if no address remap occurs. If an address remap occurs, the kind of burst depends upon the HBURST programmed in corresponding ICACHE_CRRx.

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port selected for the line refill.

- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

8.4.9 Dual-master cache

The ICACHE implements a dual-port AHB master on the main AHB bus matrix: master1 and master2 ports. This is used to split the traffic going to different destination memories.

The nonremapped traffic goes systematically to master1 port. The remapped traffic can be routed on the master2 port by programming MSTSEL in ICACHE_CRRx (on a region basis).

The code can typically be fetched as follows:

- internal flash memory and internal SRAM on master1 port (Fast bus)
- external flash memory/RAM on master2 port (Slow bus)

For systems not implementing external memories, the traffic to the internal flash memory can be decoupled from the traffic to the internal SRAM (when remapped by the ICACHE). This feature is used to prevent further processor stalls on misses.

Alongside with hit-under-miss, this dual-master feature allows the processor to have an alternative path in case of fetching from different memories.

8.4.10 ICACHE security

The ICACHE implements an Armv8-M TrustZone.

ICACHE configuration registers are protected at system level.

8.4.11 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

8.4.12 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 8.4.8*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

8.4.13 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

The code remapping at boot is not needed for a Cortex®-M33 as it implements the VTOR (vector tables) that allows a boot start address definition different than 0x0.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

8.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by fetching instructions from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 36*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves fetch performance, even if for most code execution, it is less efficient than with an n-way set associative cache mode.

8.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 92. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from the master1 or master2 port back to the execution port.

8.7 ICACHE registers

8.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
													rw	w	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

8.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

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

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

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

8.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

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

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

Bit 2 **ERRIE**: interrupt enable on cache error

Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end

Set by software to enable an interrupt generation at the end of a cache invalidate operation.

0: interrupt disabled on busy end

1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

8.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

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

Bit 2 **CERRF**: clear cache error flag

Set by software.

0: no effect

1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag

Set by software.

0: no effect

1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

8.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

8.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

8.7.7 ICACHE region x configuration register (ICACHE_CRRx)

Address offset: 0x020 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0200

Define an alias address in the code region for other regions, making them cacheable.

BASEDADDR and REMAPADDR fields are write locked (read only) when EN = 1 in ICACHE_CR.

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

Bit 31 **HBURST**: output burst type for region x
0: WRAP
1: INCR

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **MSTSEL**: AHB cache master selection for region x
0: no action (master1 selected by default)
1: master2 selected

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **REMAPADDR[31:21]**: remapped address for region x
This field replaces the alias address defined by BASEADDR field.
The only useful bits are [31:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

Bit 15 **REN**: enable for region x
0: disabled
1: enabled

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:9 **RSIZE[2:0]**: size for region x
000: reserved
001: 2 Mbytes
010: 4 Mbytes
011: 8 Mbytes
100: 16 Mbytes
101: 32 Mbytes
110: 64 Mbytes
111: 128 Mbytes

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **BASEADDR[28:21]**: base address for region x
This alias address is replaced by REMAPADDR field.
The only useful bits are [28:RI], where 21 ≤ RI ≤ 27 is the number of bits of RSIZE (see *Section 8.4.7*). If the programmed value has more LSBs, the useless bits are ignored.

8.7.8 ICACHE register map

Table 93. ICACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ICACHE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0														1	0	0
0x004	ICACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
	Reset value																														0	0	1
0x008	ICACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																														0	0	
0x00C	ICACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																														0	0	
0x010	ICACHE_HMONR	HITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ICACHE_MMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	ICACHE_CRR0	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x024	ICACHE_CRR1	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x028	ICACHE_CRR2	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0
0x02C	ICACHE_CRR3	HBURST	Res.	Res.	MSTSEL	Res.	Res.	REMAPADDR[31:21]										REN	Res.	Res.	Res.	RSIZE [2:0]			Res.	Res.	BASEADDR[28:21]						
	Reset value	0			0			0	0	0	0	0	0	0	0	0	0	0				0	0	1			0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

9 Data cache (DCACHE)

9.1 DCACHE introduction

The data cache (DCACHE) is introduced on S-AHB system bus of the Cortex®-M33 processor to improve the performance of data traffic to/from external memories.

Some specific features, like hit-under-miss and critical-word-first refill policy, optimize performance on external memories data accesses.

9.2 DCACHE main features

The main features of DCACHE are described below:

- Bus interface
  - One 32-bit AHB slave port, the system port (input from Cortex®-M33 S-AHB system interface)
  - One 32-bit AHB master port (output to main AHB bus matrix)
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for access to DCACHE registers)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Optimized cache line refill thanks to WRAP bursts of the size of the cache line (such as WRAP4 for 128-bit cache line)
  - 2-way set-associative
  - Supports both write-back and write-through policies (selectable with AHB bufferable attribute)
  - Read and write-back always allocate
  - Write-through always non-allocate (write-around)
  - Supports byte, half-word, and word writes
- Replacement and refill
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy for read transactions, minimizing processor stalls
  - Possibility to configure burst type of all AHB memory transactions: INCRw or WRAPw (size w aligned on cache line size)
- Performance counters

  The DCACHE implements four performance counters:
  - Two hit-monitor counters (32-bit): number of read hits, number of write hits
  - Two miss-monitor counters (16-bit): number of read misses, number of write misses

- Error management
  - Possibility to detect error for master port request initiated by DCACHE itself (a cache line written back into main memory, because of an eviction or a clean operation), to flag this error, and optionally to raise an interrupt
- TrustZone security support
- Maintenance operations
  - Cache invalidate: full cache invalidation, fast command, non-interruptible
  - Cache invalidate range: invalidates cache lines whose address belongs to defined range, background task, interruptible
  - Cache clean range: cleans cache lines (if dirty bit = 1, write back line, then clear dirty bit) whose address belongs to defined range, background task, interruptible
  - Cache clean and invalidate range: cleans and invalidates cache lines (if dirty bit = 1, write back line, then invalidate it) whose address belongs to defined range, background task, interruptible

9.3 DCACHE implementation

The DCACHE1 is placed on Cortex®-M33 S-AHB bus, and caches only the external RAM memory region (OCTOSPI and FMC), in the address range [0x6000 0000:0x9FFF FFFF] and [0xC000 0000:0xDFFF FFFF] of the memory map.

Indeed, by placing a bus matrix demultiplexing node in front of the DCACHE1, S-AHB bus memory requests addressing SRAM region or peripherals region (respectively in ranges [0x2000 0000:0x3FFF FFFF] and [0x4000 0000:0x5FFF FFFF]) are routed directly to the main AHB bus matrix, and the DCACHE1 is bypassed.

Table 94. DCACHE features

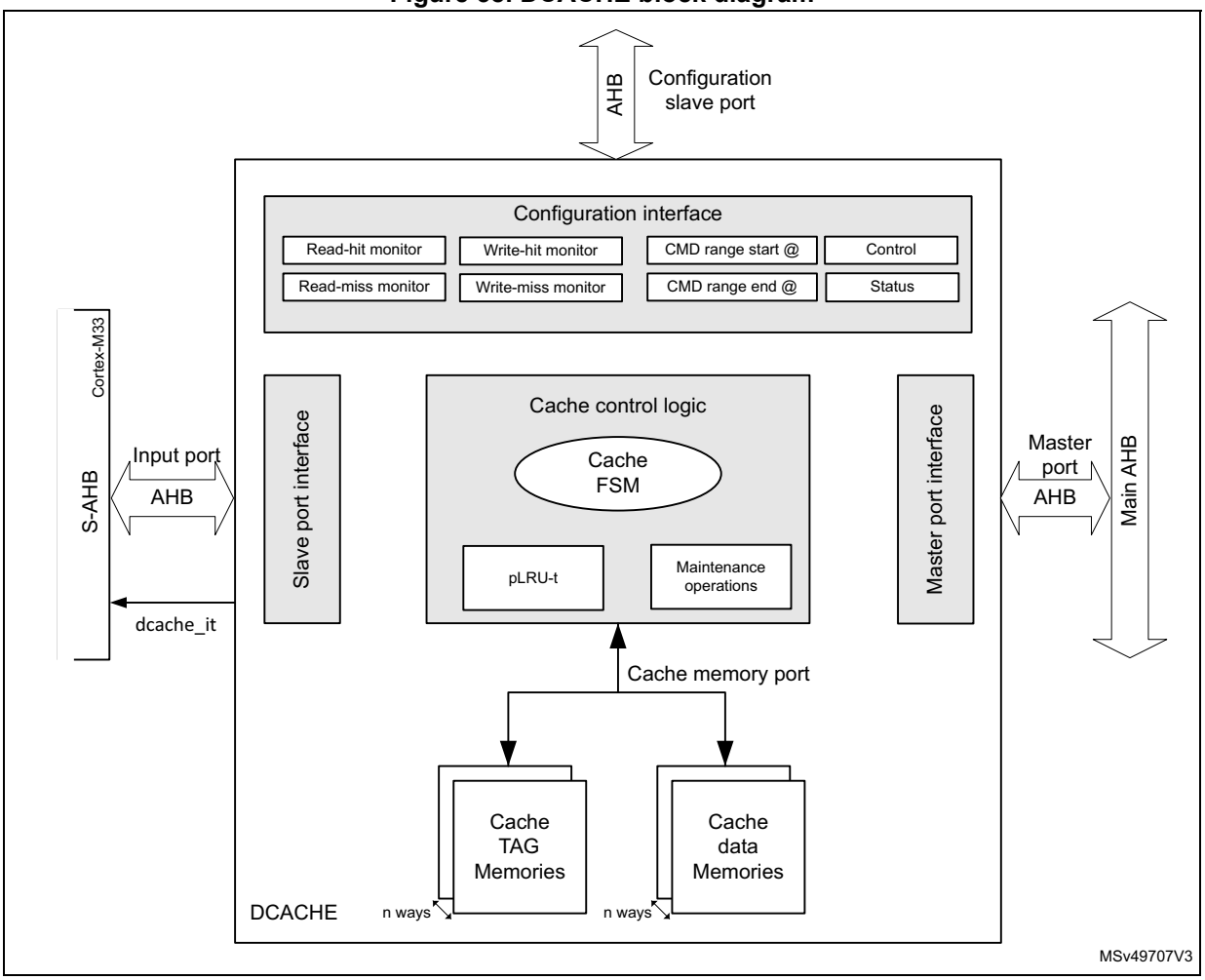
Features	DCACHE1
Number of ways	2
Cache size	4 Kbytes
Cache line width	16 bytes
Data size of AHB Master interface	32 bits

9.4 DCACHE functional description

The purpose of the data cache is to cache external memory data loads and stores, coming from the processor. These accesses include the instruction fetches that may occur at an external memory address. The DCACHE manages both read and write transactions.

9.4.1 DCACHE block diagram

Figure 38. DCACHE block diagram



9.4.2 DCACHE reset and clocks

The DCACHE is clocked on the Cortex®-M33 S-AHB bus clock.

When the DCACHE reset signal is released, a cache invalidate procedure is automatically launched, making the DCACHE busy (DCACHE_SR = 0x0000 0001).

When this procedure is finished:

- The DCACHE is invalidated: cold cache, with all cache line valid, dirty, and privilege bits = 0 (DCACHE must be filled up)
- DCACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- The DCACHE is disabled: the EN bit in DCACHE_CR holds its reset state (= 0).

Note: When disabled, the DCACHE is bypassed: slave input requests are just forwarded to the master port.

9.4.3 DCACHE TAG memory

The DCACHE TAG memory contains:

- address tags that indicate which data are contained in the cache data memories
- validity bits
- dirty bits
- privilege bits

There is one valid bit, one dirty bit, and one privilege bit per cache line (per way).

The valid bit enables/disables access to the data cache line: if the line is not valid, the data access (read or write) is performed in the main memory.

The valid bit is set when the cache line is written (refilled by either a read miss or a write-back miss).

Valid bits are reset in any of the below cases:

- after the DCACHE reset is released
- when the cache is disabled, by setting EN = 0 in DCACHE_CR (by software)
- when executing one of the DCACHE invalidate commands, setting by software CACHEINV = 0, or CACHECMD = 0b010 or 0b011 in DCACHE_CR (see *Section 9.4.8*).

The dirty bit indicates that the cache line has up-to-date values with respect to the main memory content (the cache has last right value, the main memory is not up to date).

The dirty bit is set when the cache line is written by a slave port write transaction (only in case of an access with write-back attribute).

Dirty bits are reset in any of the below cases:

- after the DCACHE reset is released
- when a line refill is performed on a read miss (on a write-back miss, the refilled cache line is modified by the written data, and dirty bit = 1)
- when the cache invalidation is performed
- when executing one of the DCACHE clean operations (cache line written back to the main memory), setting by software CACHECMD = 0b001 or 0b011 in DCACHE_CR (see *Section 9.4.8*).

The privilege bit indicates if the data is managed by a privileged entity. It is assigned according to the value of AHB privileged attribute at the input slave port, for the first access to this line (it is written only during the line refill, on read miss or write-back miss).

The privilege bit holds the same polarity as the privileged attribute: 1 for privileged access, 0 for unprivileged access.

Privilege bits are reset when the cache is invalidated, and after the DCACHE reset is released.

When a cacheable transaction is received at input slave port, its AHB address (HADDR_in) is split into the following fields (see the table below for B and W values):

- HADDR_in[B-1:0]: address byte offset, indicates which byte to select within a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) within the DCACHE

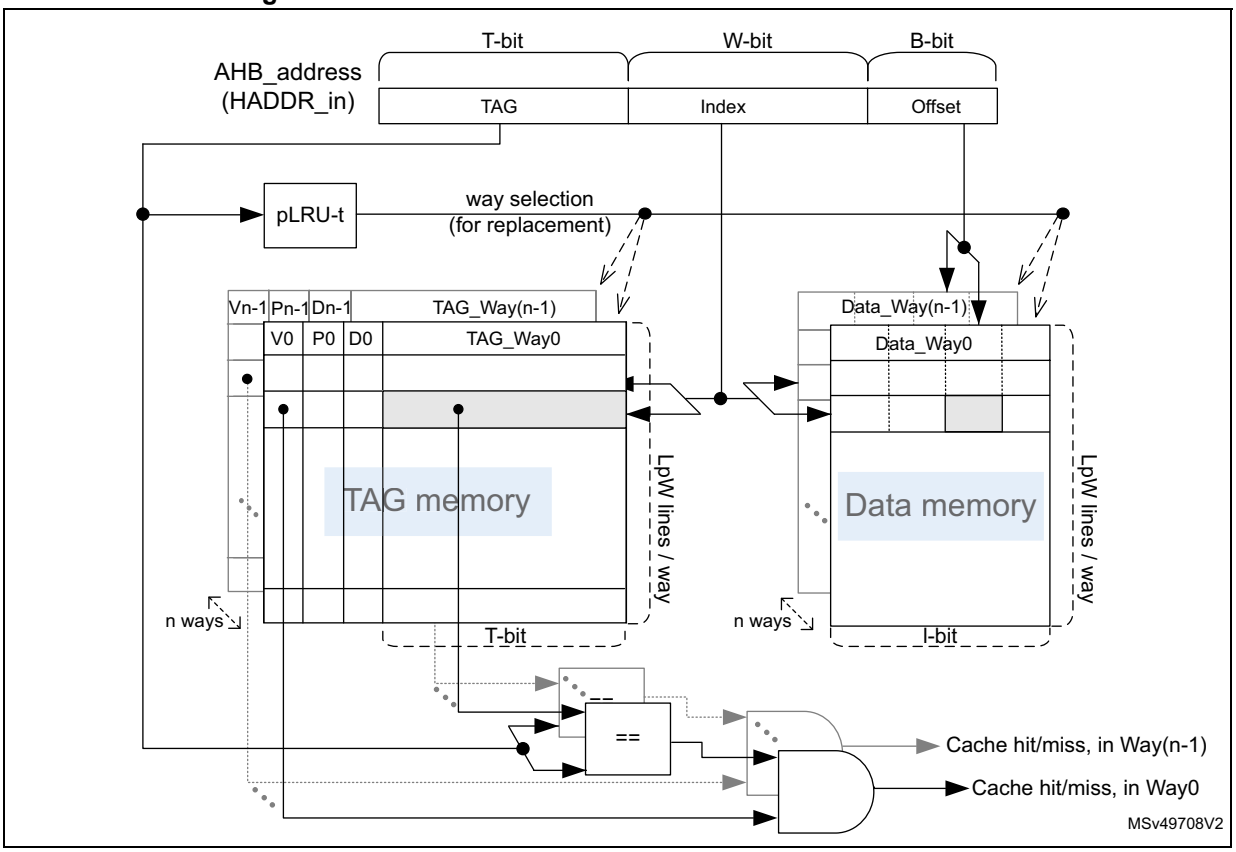
Table 95 gives DCACHE main parameters for TAG memory dimensioning.

Figure 39 shows the functional view of TAG and data memories, for an n-way set associative DCACHE.

Table 95. TAG memory dimensioning parameters

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	4 Kbytes = 4096 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines (per way)	LpW = s / (n x L) lines/way	128 lines/way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	7-bit
TAG address size	T = (32 - W - B) bit	21-bit

Figure 39. DCACHE TAG and data memories functional view



9.4.4 DCACHE enable

To activate the DCACHE functioning, the EN bit must be set in DCACHE_CR control register.

When DCACHE is disabled, it is bypassed and all transactions are copied from slave port to master port in the same clock cycle, and no comparison is performed with TAG address.

DCACHE is by default disabled at boot.

9.4.5 Cacheable and noncacheable traffic

DCACHE is developed for a Cortex®-M33 core and caches the memory regions addressed by the AHB bus connected to it.

In addition, the AHB bus traffic to the memory regions can be cacheable or noncacheable. An incoming memory request to DCACHE is defined as cacheable according to its AHB transaction memory lookup attribute.

In case of write transaction, the DCACHE write policy is defined as write-through or write-back according to its AHB transaction memory bufferable attribute (see *Table 96*).

These AHB attributes depend on the memory protection unit (MPU) programming for the addressed region.

Table 96. DCACHE cacheability for AHB transaction

AHB lookup attribute	AHB bufferable attribute	Cacheability
0	x	Read and write: non cacheable
1	0	Read: cacheable Write: (cacheable) write-through
1	1	Read: cacheable Write: (cacheable) write-back

In case of noncacheable access, the DCACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port.

The bypass does not increase the latency of the access to the targeted memory.

In case of cacheable access, the DCACHE behaves as explained in *Section 9.4.6*.

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

9.4.6 Cacheable accesses

When the DCACHE receives a cacheable transaction from Cortex®-M33 on its slave port, the DCACHE checks if the address requested is present in its TAG memory and if the corresponding cache line is valid.

For **read** transaction, there are three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache read hit**, the data is read from cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache read miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first refill policy insures minimum wait cycles for the processor, since read data can be provided while cache is still performing cache line refill (associated latency is the latency of fetching one word from main memory).

  The kind of burst generated on the DCACHE master bus depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words).

  The AHB transaction attributes are also propagated from the slave input (missing) request to the master output refill request.
- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache read hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can provide the requested read data as soon as the data is available at its master interface, thus avoiding a miss (with data fetch from main memory).

For **write-back** transaction (write transaction, with write-back bufferable attribute), there are three alternatives as well:

- The address is present inside the TAG memory, cache line is valid: **cache write-back hit**, the data is written in cache.
- The address is not present in the TAG memory (or the cache line is not valid): **cache write-back miss**.

  First, a line allocation is performed by reading the entire cache line data from main memory. The kind of burst generated on the DCACHE master bus for this line refill depends on HBURST bit in DCACHE_CR: either INCRw or WRAPw (w being the cache line width, in words), and the AHB transaction attributes are propagated from the slave port initial request.

  Once the refilled line is written in the DCACHE, the initial data provided on the slave port is written in this DCACHE line (it overwrites the data part of the cache line that was refilled just before).
- The address is not present in the TAG memory but belongs to the refill burst from main memory that is currently ongoing: **cache write-back hit** as well (hit-under-miss feature).

  Whatever the line refill is due to a read or write (missing) transaction, the DCACHE can write incoming data directly inside the refilled line, thus avoiding a miss (with refill from main memory).

For **write-through** transaction (write transaction, with write-through bufferable attribute), only two alternatives exist:

- The address is present inside the TAG memory, cache line is valid: **cache write-through hit**, the data is written both in cache and in main memory (through master port).
- The address is not present in the TAG memory (or the cache line is not valid): **cache write-through miss**, the data incoming at slave port is written only in main memory (unlike the write-back miss, there is no line allocation and data written in cache).

In case of cache refill (due to cache miss), the DCACHE selects which cache line is written with the refill data: as a 2-way set associative cache, one line among two can be used (line pointed by the address index, in each of the two ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be evicted first:

- If the dirty tag of this line equals 0 (clean data), the line is simply invalidated.
- If it equals 1 (dirty data), the line must be written back in the main memory.

  The DCACHE generates a burst write transaction on its master port, with burst type set to INCRw (w being the cache line width, in words), and with AHB memory transaction attribute signals set as below:
  - data (not instruction)
  - privileged = TAG privilege bit
  - write-back (even if it does not care)
  - normal memory
  - cacheable
  - allocate (even if it does not care)

– nonshareable

These AHB attributes cannot be propagated from the slave port (as it is the case for all other transactions emitted by the DCACHE) because the evicting transaction has no relation with the initial missing transaction. Setting of the AHB attributes is fixed, except for the privileged bit that is copied from the TAG privilege bit of the evicted line.

9.4.7 DCACHE security

The DCACHE implements an Armv8-M TrustZone.

DCACHE configuration registers are protected at system level.

9.4.8 DCACHE maintenance

The DCACHE features several maintenance operations that the software can programmed in DCACHE_CR control register:

- **Full invalidate**: invalidates the whole cache, non interruptible task.

  The software can invalidate the whole DCACHE content by programming CACHEINV in DCACHE_CR.

  When CACHEINV = 1, the DCACHE control logic sets BUSYF flag in DCACHE_SR status register, and performs the operation of cache invalidation, resetting each TAG valid bit to 0 (one valid bit per cache line). Each dirty and privilege bits are also reset to 0 during cache invalidation to prevent unknown values at next cache line validation. CACHEINV is automatically cleared.

  Once the full invalidate operation is finished, the DCACHE automatically clears BUSYF flag, and sets BSYENDF in DCACHE_SR.

  If enabled on this flag condition (BSYENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised. Then, the (empty) cache is available again.

  This full invalidate operation is not interruptible, meaning that the cache does not treat any cacheable request while BUSYF = 1. However, non-cacheable traffic is treated (since the request address is not compared to TAG ones), the DCACHE being bypassed in the same clock cycle (same behavior as when the DCACHE is disabled).

- **Invalidate range**: invalidates a certain range of addresses in the cache, background task (interruptible).

  The software can invalidate a given data region in the DCACHE by programming STARTCMD = 1 and CACHECMD = 0b010 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), the

corresponding cache line is invalidated (line TAG bits cleared, valid bit = dirty bit = privilege bit = 0).

When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the invalidate range operation. STARTCMD is also automatically cleared.

Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

During this invalidate range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the invalidation process. The TAG memory is accessed for invalidate range operation only if not already accessed by an external cache request. This implies that invalidate range execution is usually not performed in one go, but can be interrupted.

- **Clean range**: cleans a certain range of addresses in the cache, background task (interruptible).

  Cleaning a cache line means making sure that the main memory content is up-to-date with the data, which may have been modified in the cache. The clean operation consists in performing the write-back in the main memory of the cache lines that are tagged as “dirty” (the ones with TAG dirty bit set).

  The software can clean a given data region in DCACHE by programming STARTCMD = 1, and CACHECMD = 0b001 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  The DCACHE control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (DCACHE_CMDRSADDRR ≤ Line Addr ≤ DCACHE_CMDREADDRR), and the corresponding line is dirty, this line is cleaned, meaning the whole cache line is written-back in the memory through the DCACHE master port, and its TAG dirty bit is cleared.

  When STARTCMD is set, the DCACHE control logic sets BUSYCMDF in DCACHE_SR and launches the clean range operation. STARTCMD is also automatically cleared.

  Once the operation is finished (all TAG memory parsed), the DCACHE automatically clears BUSYCMDF and sets CMDENDF in DCACHE_SR.

  If enabled on this flag condition (CMDENDIE = 1 in DCACHE_IER), the DCACHE interrupt is raised.

  During this clean range operation, the DCACHE is interruptible, meaning it can accept new incoming requests that take higher priority than the cleaning process. The TAG memory is accessed for clean range operation only if not already accessed by an external cache request. This implies that clean range execution is usually not performed in one go, but can be interrupted.

  It is under the software responsibility that no bus initiator attempts to change the content of the region being cleaned until clean range is completed. For that, the software can take advantage of BUSYCMDF flag in DCACHE_SR, and can poll this flag to prevent any spurious access to the area being cleaned.

Alternatively it can also rely on the command end flag (CMDENDF) or on the DCACHE interrupt to detect the end of the clean range execution.

- **Clean and invalidate range**: cleans and invalidates a certain range of addresses in the cache, background task (interruptible).

  This operation cleans the “dirty” cache lines that belong to the operation address range (the same as the clean range operation), and also invalidates all the (valid) cache lines that belong to this address range (whether they are dirty or not).

  The software can launch this clean and invalidate range operation, by programming STARTCMD = 1, and CACHECMD = 0b011 in DCACHE_CR, after the address range was programmed into DCACHE_CMDRSADDRR (range start address) and DCACHE_CMDREADDRR (range end address).

  This sets and clears the same flags, and potentially the same interrupt as invalidate range or clean range operations.

9.4.9 DCACHE performance monitoring

The DCACHE provides the following monitors for performance analysis:

- The two 32-bit read-hit and write-hit monitors count the AHB transactions at the DCACHE input (slave port) that do not generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is present in the TAG memory, or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 9.4.6*).
- The two 16-bit read-miss and write-miss monitors count the AHB transactions at the DCACHE input (slave port) that generate a transaction on the DCACHE output (master port).

  These monitors also take into account all accesses whose address is not present neither in the TAG memory, nor in the refill buffer.

Upon reaching their maximum values, the monitors do not wrap over.

The software can perform the following tasks:

- Enable/stop the read (write) hit monitor, through R(W)HITMEN in DCACHE_CR.
- Reset the read (write) hit monitor, by setting R(W)HITMRST in DCACHE_CR.
- Enable/stop the read (write) miss monitor, through R(W)MISSMEN in DCACHE_CR.
- Reset the read (write) miss monitor, by setting R(W)MISSMRST in DCACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

9.4.10 DCACHE boot

The DCACHE is disabled (EN = 0 in DCACHE_CR) at boot.

Once the boot is finished, the DCACHE can be enabled (software setting EN = 1 in DCACHE_CR).

9.5 DCACHE low-power modes

At product level, the DCACHE reduces the power consumption by loading/storing data from/to the internal DCACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even much higher, since the cached main memories are external.

9.6 DCACHE error management and interrupts

A transaction initiated on the DCACHE master port may return an error (a write attempt into a read-only memory, for instance). If the master port request was propagated from a slave port request, the error is propagated back to the slave port. If ever the master port request is initiated by the DCACHE itself (a cache line is written back into the main memory because of an eviction or a clean operation), the DCACHE receives this functional error and flags it internally by setting the ERRF flag in DCACHE_SR.

In such a case, an interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in DCACHE_IER).

Another case of potential interrupt generation is at the end of a full invalidate operation: when the cache busy state is finished, the DCACHE sets BSYENDF flag in DCACHE_SR.

An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in DCACHE_IER).

Last case is at the end of invalidate and/or clean range operations: when the command busy state is finished, the DCACHE sets CMDENDF flag in DCACHE_SR.

An interrupt is also generated if the corresponding interrupt enable bit is set (CMDENDIE = 1 in DCACHE_IER).

All DCACHE interrupt sources raise the same and unique interrupt signal, dcache_it, and then use the same interrupt vector.

Table 97. DCACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
DCACHE	Functional error	ERRF in DCACHE_SR	ERRIE in DCACHE_IER	Set CERRF to 1 in DCACHE_FCR
	End of busy state (full invalidate finished)	BSYENDF in DCACHE_SR	BSYENDIE in DCACHE_IER	Set CBSYENDF to 1 in DCACHE_FCR
	End of cache operations (address range based)	CMDENDF in DCACHE_SR	CMDENDIE in DCACHE_IER	Set CCMDENDF to 1 in DCACHE_FCR

The DCACHE also propagates all AHB bus errors (such as security issues, address decoding issues) from master port back to the S-AHB slave port.

9.7 DCACHE registers

9.7.1 DCACHE control register (DCACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	STARTCMD	CACHECMD[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
				w	rw	rw	rw							w	rw

Bit 31 **HBURST:** output burst type for cache master port read accesses
Can be set by software, only when EN = 0.
Master port write accesses are always done in INCR burst type.
0: WRAP
1: INCR

Bits 30:24 Reserved, must be kept at reset value.

Bit 23 **WMISSMRST:** write-miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache write-miss monitor

Bit 22 **WHITMRST:** write-hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache write-hit monitor

Bit 21 **WMISSMEN:** write-miss monitor enable
0: cache write-miss monitor switched off. Stopping the monitor does not reset it.
1: cache write-miss monitor enabled

Bit 20 **WHITMEN:** write-hit monitor enable
0: cache write-hit monitor switched off. Stopping the monitor does not reset it.
1: cache write-hit monitor enabled

Bit 19 **RMISSMRST:** read-miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache read-miss monitor

Bit 18 **RHITMRST:** read-hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache read-hit monitor

Bit 17 **RMISSMEN:** read-miss monitor enable
0: cache read-miss monitor switched off. Stopping the monitor does not reset it.
1: cache read-miss monitor enabled

Bit 16 **RHITMEN:** read-hit monitor enable
0: cache read-hit monitor switched off. Stopping the monitor does not reset it.
1: cache read-hit monitor enabled

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

Bit 11 **STARTCMD:** starts maintenance command (maintenance operation defined in CACHECMD).
Can be set by software, only when EN = 1, BUSYCMDF = 0, BUSYF = 0 and CACHECMD = 0b001, 0b010 or 0b011.
Cleared by hardware when the BUSYCMDF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: command operation (cache maintenance) finished
1: start maintenance command (cache maintenance)

Bits 10:8 **CACHECMD[2:0]:** cache command maintenance operation (cleans and/or invalidates an address range)
Can be set and cleared by software, only when no maintenance command is ongoing (BUSYCMDF = 0).
000: no operation
001: clean range
010: invalidate range
011: clean and invalidate range
others: reserved

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

Bit 1 **CACHEINV:** full cache invalidation
Can be set by software, only when EN = 1.
Cleared by hardware when the BUSYF flag is set (during full cache invalidation operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN:** enable
0: cache disabled
1: cache enabled

9.7.2 DCACHE status register (DCACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

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

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

Bit 4 **CMDENDF:** command end flag
Cleared by writing DCACHE_FCR.CCMDENDF = 1.
0: cache busy or in idle
1: CACHECMD command finished

Bit 3 **BUSYCMDF:** command busy flag
0: cache not busy on a CACHECMD command
1: cache busy on a CACHECMD command (clean and/or invalidate an address range)

Bit 2 **ERRF:** cache error flag
Cleared by writing DCACHE_FCR.CERRF = 1.
0: no error
1: an error occurred during the operation (eviction or clean operation write-back error).

Bit 1 **BSYENDF:** full invalidate busy end flag
Cleared by writing DCACHE_FCR.CBSYENDF = 1.
0: cache busy or in idle
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF:** full invalidate busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

9.7.3 DCACHE interrupt enable register (DCACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

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

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

Bit 4 **CMDENDIE**: interrupt enable on command end
Set by software to enable an interrupt generation at the end of a cache command (clean and/or invalidate an address range)
0: interrupt disabled on command end
1: interrupt enabled on command end

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ERRIE:** interrupt enable on cache error
Set by software to enable an interrupt generation in case of cache functional error (eviction or clean operation write-back error)
0: interrupt disabled on error
1: interrupt enabled on error

Bit 1 **BSYENDIE:** interrupt enable on busy end
Set by SW to enable an interrupt generation at the end of a cache full invalidate operation.
0: Interrupt disabled on busy end
1: Interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

9.7.4 DCACHE flag clear register (DCACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

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

Bit 4 **CCMDENDF:** clear command end flag
Set by software.
0: no effect
1: clears CMDENDF flag in DCACHE_SR

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CERRF:** clear cache error flag
Set by software.
0: no effect
1: clears ERRF flag in DCACHE_SR

Bit 1 **CBSYENDF:** clear full invalidate busy end flag
Set by software.
0: no effect
1: clears BSYENDF flag in DCACHE_SR

Bit 0 Reserved, must be kept at reset value.

9.7.5 DCACHE read-hit monitor register (DCACHE_RHMONR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **RHITMON[31:0]:** cache read-hit monitor counter

9.7.6 DCACHE read-miss monitor register (DCACHE_RMMONR)

Address offset: 0x014

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RMISSMON[15:0]:** cache read-miss monitor counter

9.7.7 DCACHE write-hit monitor register (DCACHE_WHMONR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WHITMON[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WHITMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WHITMON[31:0]:** cache write-hit monitor counter

9.7.8 DCACHE write-miss monitor register (DCACHE_WMMONR)

Address offset: 0x024

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WMISSMON[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WMISSMON[15:0]:** cache write-miss monitor counter

9.7.9 DCACHE command range start address register (DCACHE_CMDRSADDRR)

Address offset: 0x028

Reset value: 0x0000 0000

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

Bits 31:4 **CMDSTARTADDR[31:4]:** start address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

Bits 3:0 Reserved, must be kept at reset value.

9.7.10 DCACHE command range end address register (DCACHE_CMDREADDRR)

Address offset: 0x02C

Reset value: 0x0000 0000

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

Bits 31:4 **CMDENDADDR[31:4]:** end address of range to which the cache maintenance command specified in DCACHE_CR.CACHECMD field applies

This register must be set before DCACHE_CR.CACHECMD is written.

Bits 3:0 Reserved, must be kept at reset value.

9.7.11 DCACHE register map

Table 98. DCACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DCACHE_CR	HBURST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN	Res.	Res.	Res.	Res.	STARTCMD	CACHECMD[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
	Reset value	0								0	0	0	0	0	0	0	0					0	0	0	0							0	0

Table 98. DCACHE register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	DCACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
	Reset value																												0	0	0	0	1
0x008	DCACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
	Reset value																												0		0	0	
0x00C	DCACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
	Reset value																												0		0	0	
0x010	DCACHE_ RHMONR	RHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DCACHE_ RMMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	DCACHE_ WHMONR	WHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DCACHE_ WMMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WMISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DCACHE_ CMDRSADDRR	CMDSTARTADDR[31:4]																											Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x02C	DCACHE_ CMDREADDRR	CMDENDADDR[31:4]																											Res.	Res.	Res.	Res.	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Refer to *Section 2.3* for the register boundary addresses.

10 Power control (PWR)

10.1 Introduction

The power controller manages the device power supplies and power modes transitions.

10.2 PWR main features

The power controller (PWR) main features are:

- Power supplies and supply domains
  - Core domain (V_{CORE})
  - V_{DD} domain
  - Backup domain (V_{SW})
  - Analog domain (V_{DDA})
  - Supply for the SMPS power stage (available on SMPS packages)
  - V_{DDIO2} supply for ten I/Os (PD6, PD7, PG9:14, PB8, PB9)
  - V_{DDUSB} for USB transceiver
- System supply voltage regulation
  - SMPS step-down converter (STM32H563/573xx devices only)
  - Linear voltage regulator (LDO)
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - Out of functional range temperature monitor
  - Out of functional range Backup domain voltage monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes
- V_{BAT} battery charging
- TrustZone security and privileged protection

10.3 PWR pins and internal signals

Table 99. PWR input/output pins

Pin name	Signal type	Description
VDD	Supply	Main supply
GND	Supply	Main ground
VDDA	Supply	Analog peripherals supply
VSSA	Supply	Analog peripherals ground
VDDIO2	Supply	Independent I/O supply
VDDUSB	Supply	USB supply
VCAP	Supply	Logic supply (V_{CORE})
VBAT	Supply	Backup domain supply
VDDSMPS(1)	Supply	SMPS supply
VSSSMPS(1)	Supply	SMPS ground
VLXSMPS(1)	Supply	SMPS output
VREF+	Supply	ADC/DAC high reference voltage
VREF-	Supply	ADC/DAC low reference voltage
WKUPx (x = 1 to 8)	Input	Wake-up pins
CSLEEP	Output	MCU in Sleep mode
CDSTOP	Output	CPU in Stop modes

1. STM32H563/573xx devices only.

Table 100. PWR internal input/output signals

Internal signal name	Signal type	Description
WKUPx (x = 1 to 8)	Input	Wake-up event source

10.4 PWR power supplies and supply domains

Figure 40. Power supply with SMPS (STM32H563/573xx devices only)

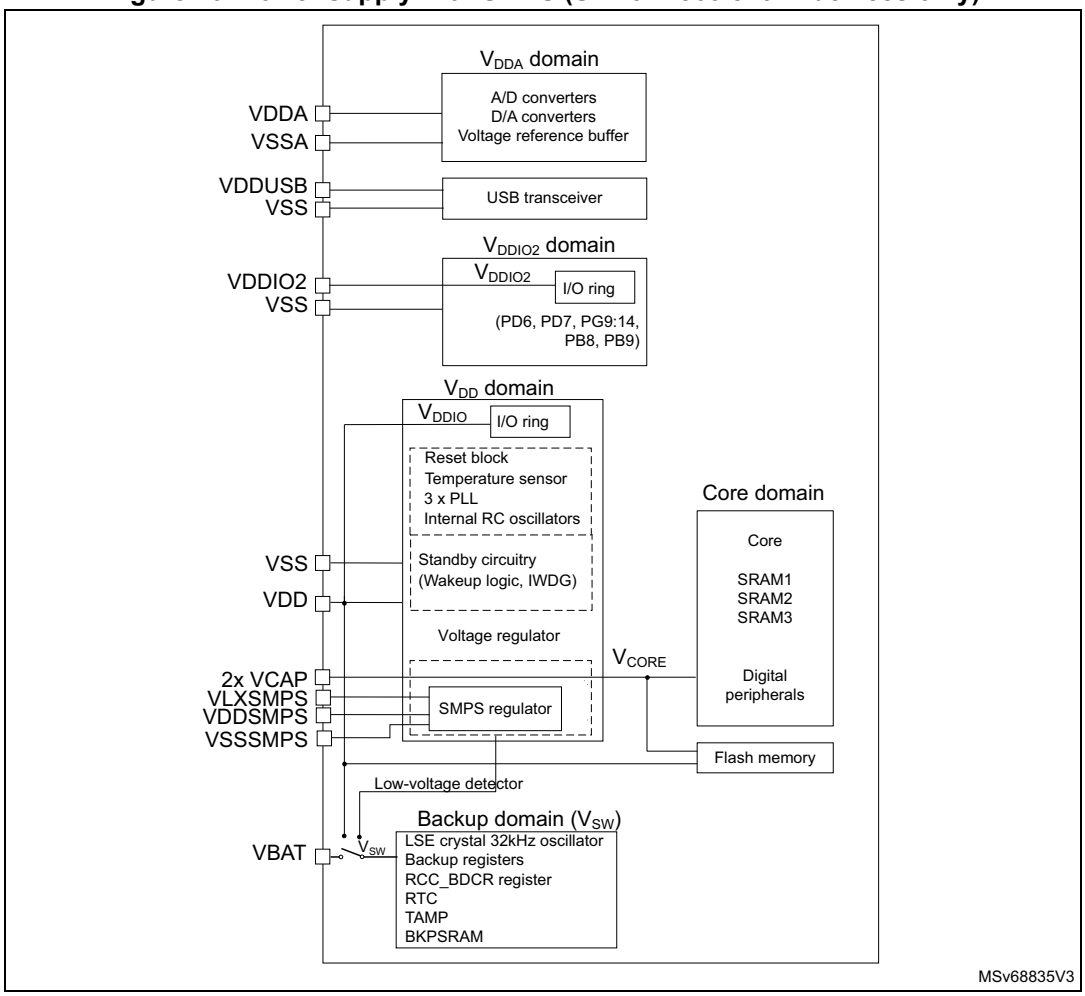
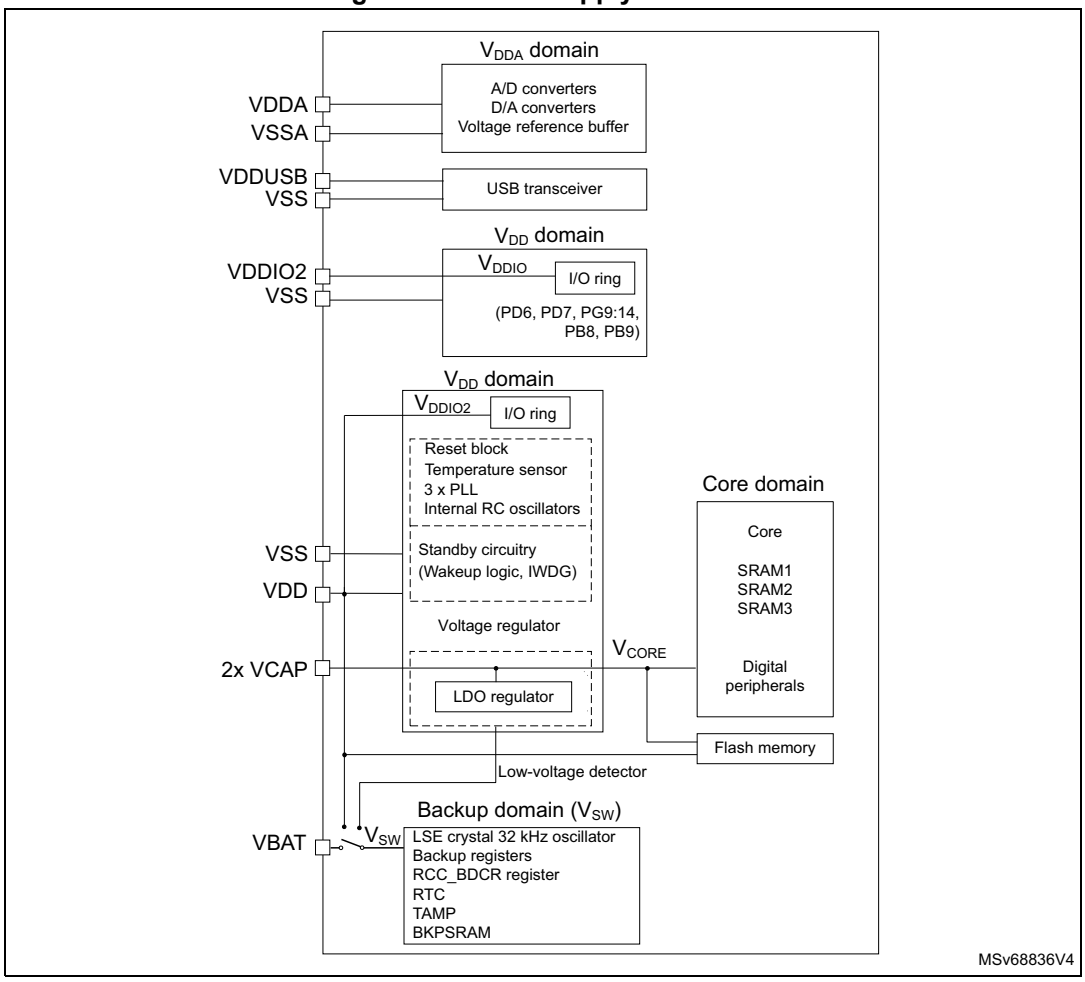


Figure 41. Power supply with LDO



10.4.1 External power supplies

The devices require a 1.71 to 3.6 V V_{DD} operating voltage supply. Several independent supplies can be provided for specific peripherals, but must not be provided without a valid operating supply on the VDD pin:

- V_{DD} = 1.71 to 3.6 V

  V_{DD} is the external power supply for the I/Os, the internal regulator, and the system analog (such as reset, power management, and internal clocks). It is provided externally through the VDD pins.

- V_{DDA} = 1.62 V (ADCs, DACs) / 2.1 (VREFBUF) to 3.6 V

  V_{DDA} is the external analog power supply for A/D converters, D/A converters, and voltage reference buffer. The V_{DDA} voltage level is independent from the V_{DD} voltage, and must preferably be connected to V_{DD} when these peripherals are not used.

- V_{DDSMPS} = 1.71 to 3.6 V

  V_{DDSMPS} is the external power supply for the SMPS step-down converter. It is provided externally through the VDDSMPS supply pin, and must be connected to the same supply as the VDD pin.

- V_{LXSMPS} is the switched SMPS step-down converter output.

Note: The SMPS power supply pins are available only on specific packages, with SMPS step-down converter option.

- V_{DDUSB} = 3.0 to 3.6 V

  V_{DDUSB} is the external independent power supply for USB transceivers. The V_{DDUSB} voltage level is independent from the V_{DD} voltage and must preferably be connected to V_{DD} when the USB is not used.
- V_{DDIO2} = 1.08 to 3.6 V

  V_{DDIO2} is the external power supply for ten I/Os (PD6, PD7, PG9:14, PB8, PB9). The V_{DDIO2} voltage level is independent from the V_{DD} voltage and must preferably be connected to V_{DD} when those pins are not used.
- V_{CAP} = 1.0 to 1.35 V: digital core domain supply

  This power supply is independent from all the other power supplies:
  - When the voltage regulator is enabled, V_{CORE} is delivered by the internal voltage regulator.
  - When the voltage regulator is disabled, V_{CORE} is delivered by an external power supply through VCAP pin, or by the SMPS
- V_{BAT} = 1.2 to 3.6 V

  V_{BAT} is the power supply when V_{DD} is not present (through power switch) for RTC, external clock 32 kHz oscillator, backup registers, and, optionally, backup SRAM.
- V_{REF-}, V_{REF+}

  V_{REF+} is the input reference voltage for ADCs and DACs. It is also the output of the internal voltage reference buffer when enabled.

  V_{REF+} can be grounded when ADC and DAC are not active.

  The internal voltage reference buffer supports four output voltages, configured with VRS bit in the VREFBUF_CSR register:
  - V_{REF+} ~ 1.8 V. (requires $V_{DDA} \geq 2.1$ V)
  - V_{REF+} ~ 2.048 V (requires $V_{DDA} \geq 2.4$ V)
  - V_{REF+} ~ 2.5 V (requires $V_{DDA} \geq 2.8$ V)

  VREF- and VREF+ pins are not available on all packages. When not available, they are bonded to VSSA and VDDA, respectively.

  When the VREF+ is double-bonded with VDDA in a package, the internal voltage reference buffer is not available and must be kept disabled.

  V_{REF-} must always be equal to V_{SSA}.

10.4.2 Internal regulators

The devices embed two regulators: one LDO and one SMPS[a] to provide the V_{CORE} supply for digital peripherals, SRAMs (except BKPSRAM), and embedded flash memory.

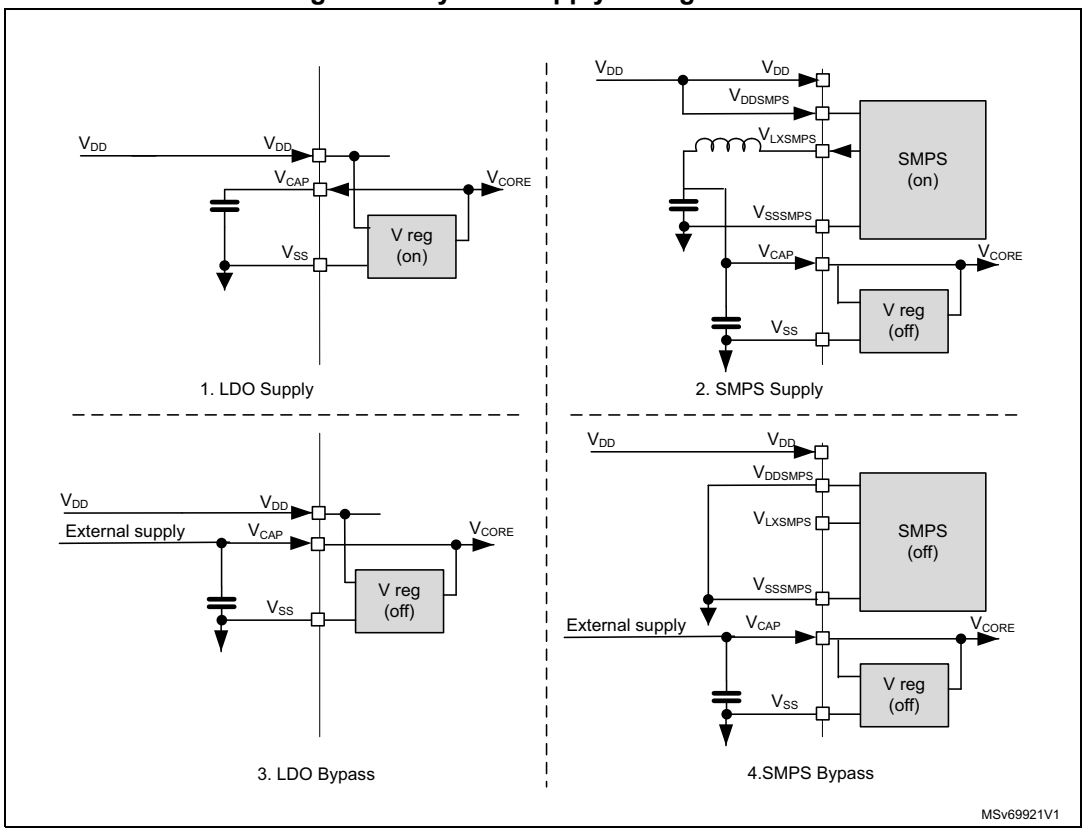
The SMPS generates this voltage on VCAP (two pins), with a total external capacitor of 10 µF (typical), and requires an external coil of 2.2 µH (typical). The LDO generates this voltage on VCAP, with a total external capacitance of 4.7 µF (typical).

Both regulators can provide four different voltages (voltage scaling), and can operate in Stop modes.

a. STM32H563/573xx devices only.

Depending on the package configuration (SMPS or LDO), the regulator is selected by hardware. SMPS and LDO regulator are exclusively selected. The supply configurations shown in *Figure 38* are supported for the V_{CORE} domain supply.

Figure 42. System supply configurations



Startup with V_{CORE} provided from an external supply (Bypass)

When supplied in Bypass mode, V_{CORE} must first settle at default level (≥ 1.1 V), before V_{DD} reaches POR threshold level.

Due to the LDO default state after power-up (enabled by default), the external V_{CORE} voltage must remain higher than 1.1 V until the LDO is disabled by software. When the LDO is disabled, the external V_{CORE} voltage can be adjusted according to the user application needs (refer to section *General operating conditions* of the datasheet for details on V_{CORE} level versus the maximum operating frequency).

When operating in Bypass mode, the application must adjust VOS, using bits VOS[1:0] in PWR_VOSCR register. VOS[1:0] must be set according to the external provided core voltage level and related performance.

To adjust the VOS level, the software must select sequentially the intermediate levels.

- When increasing the performance:
  - First, voltage scaling must be incremented (for example when changing from VOS3 to VOS0, lower levels must be selected in the VOS[1:0] bits: VOS2, VOS1, and then VOS0).
  - The external voltage can be increased
  - The system frequency can be increased
- When decreasing the performance:
  - The system frequency must be decreased
  - The external voltage must be decreased
  - The voltage scaling can be decremented (for example when changing from VOS1 to VOS3, lower levels must be selected in the VOS[1:0] bits: VOS2, and then VOS3)

10.4.3 Power-up and power-down power sequences

During power-up and power-down phases, the following power sequence requirements must be respected:

- When V_{DD} is below 1 V, other power supplies (V_{DDA}, V_{DDIO2}, V_{DDUSB}) must remain below V_{DD} + 300 mV.
- When V_{DD} is above 1 V, all power supplies are independent.

During the power-down phase, V_{DD} can temporarily become lower than other supplies only if the energy provided to the MCU remains below 1 mJ. This allows external decoupling capacitors to be discharged with different time constants during the power-down transient phase.

10.4.4 Independent analog peripherals supply

To improve ADC and DAC conversion accuracy and to extend the supply flexibility, the analog peripherals have an independent power supply that can be separately filtered and shielded from noise on the PCB:

- The voltage supply input of the analog peripherals is available on a separate VDDA pin.
- An isolated supply ground connection is provided on the VSSA pin.

The V_{DDA} supply voltage can be different from V_{DD}. The presence of V_{DDA} must be checked before enabling any of the analog peripherals supplied by V_{DDA} (A/D converter, D/A converter, voltage reference buffer).

Power supply level monitoring is available on VDDA via AVDO bit in PWR_VMSR register.

When a single supply is used, V_{DDA} can be externally connected to V_{DD} through the external filtering circuit to ensure a noise-free V_{DDA} reference voltage.

ADC and DAC reference voltage

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect to VREF+, a separate reference voltage lower than V_{DDA}. V_{REF+} is the highest voltage, represented by the full scale value, for an analog input (ADC) or output (DAC) signal.

V_{REF+} can be provided either by an external reference of by an internal buffered voltage reference (VREFBUF). The internal voltage reference can output a configurable voltage,

namely 1.8, 2.048 or 2.4 V. The internal voltage reference can also provide the voltage to external components through VREF+ pin. Refer to the device datasheet and to *Section 16: Voltage reference buffer (VREFBUF)* for further information.

Note: The VREF+ and VREF- pins are not available on all packages (internally connected, respectively, to VDDA and VSSA). Do not enable the internal voltage reference buffer when an external power supply is applied to the VREF+ pin.

10.4.5 Independent I/O supply rail

Some I/Os (PD6, PD7, PG9:14, PB8, PB9) are supplied from a separate supply rail. The power supply for this rail can range from 1.08 to 3.6 V, and is provided externally through the VDDIO2 pin. The V_{DDIO2} voltage level is completely independent from V_{DD} or V_{DDA}. The VDDIO2 pin is available only for some packages. Refer to the pinout diagrams or tables in the related device datasheet(s) for the I/O list(s).

Power supply level monitoring is available on VDDIO2 via VDDIO2RDY bit in PWR_VMSR register.

10.4.6 Independent USB transceivers supply

The USB transceivers are supplied from a separate VDDUSB power supply pin. V_{DDUSB} range is from 3.0 to 3.6 V, and is completely independent from V_{DD} or V_{DDA}.

Power supply level monitoring is available on VDDUSB via the USB33RDY bit in PWR_VMSR register.

Before setting USB33SV bit in PWR USB supply control register (PWR_USBSCR), check that V_{DDUSB} is available by monitoring USB33RDY bit in PWR voltage monitor status register (PWR_VMSR). The $V_{DD33USB}$ supply level detector must be enabled through USB33DEN bit in PWR USB supply control register (PWR_USBSCR).

Setting USB33SV bit is mandatory to use the USB peripheral. It is used to validate the V_{DDUSB} supply for electrical and logical isolation purposes.

10.4.7 Backup domain

To retain the content of the backup registers and to supply the RTC function when V_{DD} is turned off, the VBAT pin can be connected to an optional backup voltage supplied by a battery or by another source.

The VBAT pin powers the RTC unit, the LSE oscillator, PI8[a], and PC13 to PC15 I/Os, allowing the RTC to operate even when the main power supply is turned off. The backup SRAM is optionally powered by VBAT pin when the BREN bit is set in the *PWR Backup domain control register (PWR_BDCR)*. The switch to the V_{BAT} supply is controlled by the power-down reset embedded in the Reset block.

Warning: During $t_{RSTTEMPO}$ (temporization at V_{DD} startup) or after a PDR has been detected, the power switch between V_{BAT} and V_{DD} remains connected to V_{BAT}.
During the startup phase, if V_{DD} is established in less than

a. Not available on all devices, check the datasheet.

$t_{RSTTEMPO}$ (refer to the datasheet for its value) and $V_{DD} > V_{BAT}$ + 0.6 V, a current may be injected into V_{BAT} through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

If no external battery is used in the application, it is recommended to connect V_{BAT} externally to V_{DD} with a 100 nF external ceramic decoupling capacitor.

When the Backup domain is supplied by V_{BAT} (analog switch connected to V_{BAT}), the following pins are available:

- PC13, PI8[a], PC14, and PC15, which can be configured by RTC or LSE (refer to *Section 30.3: RTC functional description*)
- PC13, PI8[a], PA0, PA1, and PA2 when they are configured by TAMP peripheral as tamper pins.

Note: The analog switch can transfer only a limited amount of current, hence the use of GPIOs PC13 to PC15 and PI8[a] in output mode is restricted: the frequency must be limited (check the datasheet) with a maximum load of 30 pF, and these I/Os cannot be used as a current source (for example to drive a LED).

Backup domain access

After a system reset, the Backup domain (RCC Backup domain control register RCC_BDCR, RTC registers, TAMP registers, backup registers, and backup SRAM) is protected against possible unwanted write accesses. To enable access to the Backup domain, set the DBP bit in the *PWR Backup domain control register (PWR_BDCR)* to enable access to the Backup domain.

Backup RAM

The Backup domain includes up to 4 Kbytes of backup RAM accessible in 32-, 16-, or 8-bit data mode. The backup RAM is supplied from the backup regulator in the Backup domain. When the backup regulator is enabled through BREN bit in the PWR_BDCR, the backup RAM content is retained in Standby and/or VBAT mode (it can be considered as an internal EEPROM if VBAT is always present).

The backup regulator can be ON or OFF, depending on if the application needs the backup RAM function in Standby or VBAT modes.

The backup RAM is read protected and mass erased when a tamper event occurs, this is to prevent confidential data (such as a cryptographic private key) from being accessed.

The backup RAM can be erased in the following ways:

- through the flash interface after a full product state regression
- after a tamper event
- after a Backup domain reset

a. Not available on all devices, check the datasheet.

V_{BAT} battery charging

When V_{DD} is present, it is possible to charge the external battery on VBAT through an internal resistance.

The V_{BAT} charging is done either through a 5 or a 1.5 kΩ resistor, depending upon the VBRS bit value in the PWR_BDCR register.

The battery charging is enabled by setting VBE bit in the PWR_BDCR register. It is automatically disabled in V_{BAT} mode.

10.5 PWR system supply voltage regulation

10.5.1 SMPS and LDO embedded regulators

The devices embed two internal regulators[a], exclusively enabled by hardware on power-on reset, depending upon package configuration. To supply the V_{CORE} from an external source, it is possible to disable the regulator by setting the BYPASS bit in the PWR_SCCR register.

The BYPASS bit is written once after power-on reset. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

When V_{CORE} is supplied from an external source the externally applied voltage level must be reflected in the VOSx bits in the PWR_VOSCR register refer to *Startup with VCORE provided from an external supply (Bypass)*.

Both regulators can provide four different voltages (voltage scaling) and can operate in Stop modes.

10.5.2 V_{CORE} supply versus reset, voltage scaling, and low-power modes

After reset, the V_{CORE} is in VOS3.

When exiting the Stop or Standby mode, the voltage range is the VOS3.

10.5.3 Embedded voltage regulator operating modes

There are three different power modes: Run, Stop, and Standby modes.

Run mode

The voltage regulator provides full power to the V_{CORE} domain (core, memories, and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS0, VOS1, VOS2, and VOS3) that are configured through the VOS bits in the PWR voltage scaling control register (PWR_VOSCR).

The VOS voltage scaling allows optimization of the power consumption when the system is clocked below the maximum frequency. By default, VOS3 is selected after system reset.

VOSx bits can be changed on-the-fly to adapt to the required system performance.

a. STM32H563/573xx devices only.

Stop mode

The voltage regulator supplies the V_{CORE} domain to retain the content of registers and internal memories. The regulator mode is selected through the SVOS bits in the PWR power mode control register (PWR_PMCR).

Stop mode power consumption can be further reduced using SVO4 (lower voltage level than VOS3) and even further with SVOS5.

Standby mode

The regulator is OFF and the V_{CORE} domains are powered down. The content of the registers and memories is lost except for the Standby circuitry and the Backup domain.

10.6 PWR power supply and temperature supervision

Power supply level monitoring is available on the following supplies:

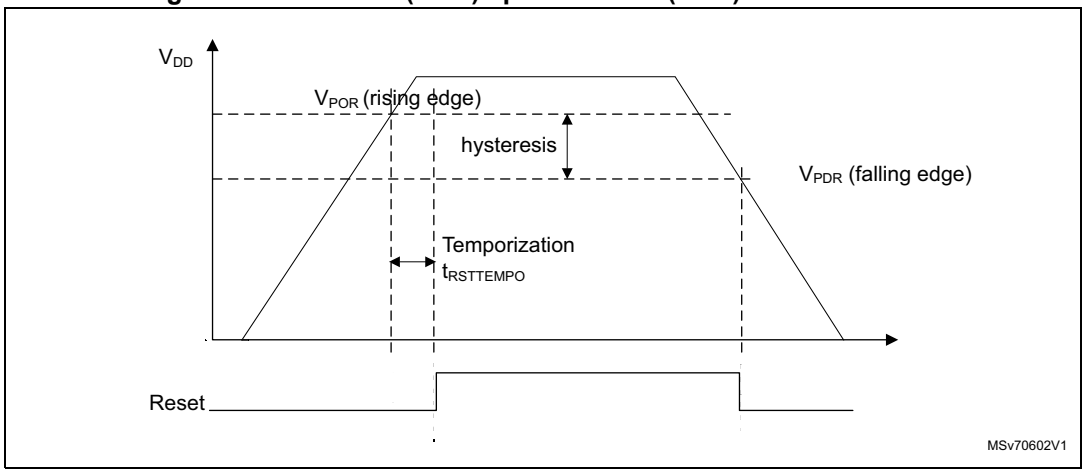
- V_{DD} via POR/PDR (see *Section 10.6.1*), BOR (see *Section 10.6.2*), and PVD monitor (see *Section 10.6.3*)
- V_{DDA} via AVD monitor see (*Section 10.6.4*)
- V_{BAT} via V_{BAT} threshold (see *Section 10.6.6*)
- V_{DDIO2} via VDDIO2RDY bit (see *Section 10.11.10*)
- Temperature monitoring (see *Section 10.6.7*)

10.6.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in reset mode when V_{DD} is below a specified VPOR threshold, without the need for an external reset circuit. Once the supply level is above the VPOR threshold, the system is taken out of reset (see *Figure 39*). For more details concerning the reset thresholds refer to the electrical characteristics section of the datasheets.

Figure 43. Power-on (POR) / power-down (PDR) reset waveform



1. For thresholds and hysteresis values refer to the datasheets.

10.6.2 Brownout reset (BOR)

During power-on, the brownout reset (BOR) keeps the system under reset until the V_{DD} supply voltage reaches the specified V_{BOR} threshold.

The V_{BOR} threshold is configured through system option bytes:

- BOR OFF (BORH_EN = 0)
- BOR level 1 (V_{BOR1})
- BOR level 2 (V_{BOR2})
- BOR level 3 (V_{BOR3})

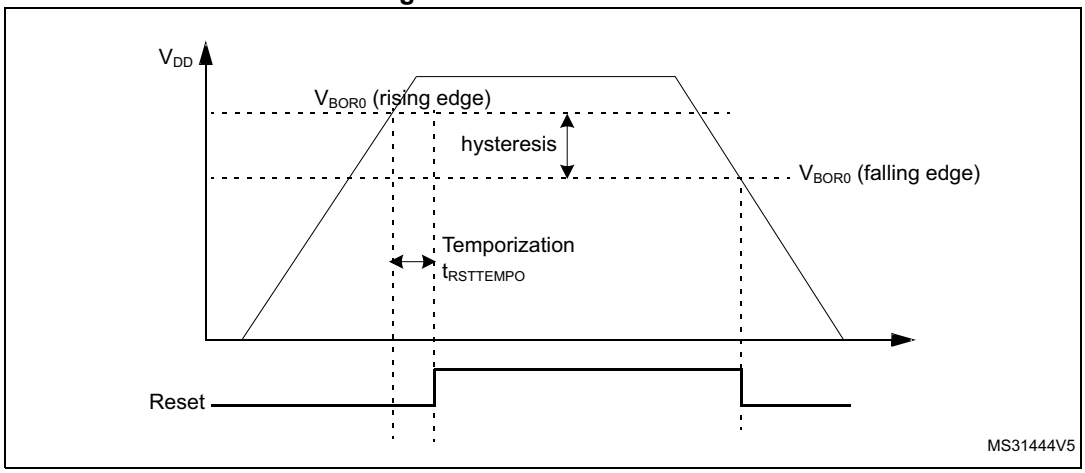
By default, BOR is OFF, it can be enabled by setting BORH_EN option bit.

For more details on the brownout reset thresholds, refer to the section "*Electrical characteristics*" of the product datasheets.

A system reset is generated when the BOR is enabled and V_{DD} supply voltage drops below the selected V_{BOR} threshold.

BOR can be disabled by programming the BORH_EN option bit to 0. To disable the BOR function, V_{DD} must have been higher than the POR threshold to start the system option byte programming sequence. Once BOR is disabled, the power-down is monitored by the PDR.

Figure 44. BOR thresholds



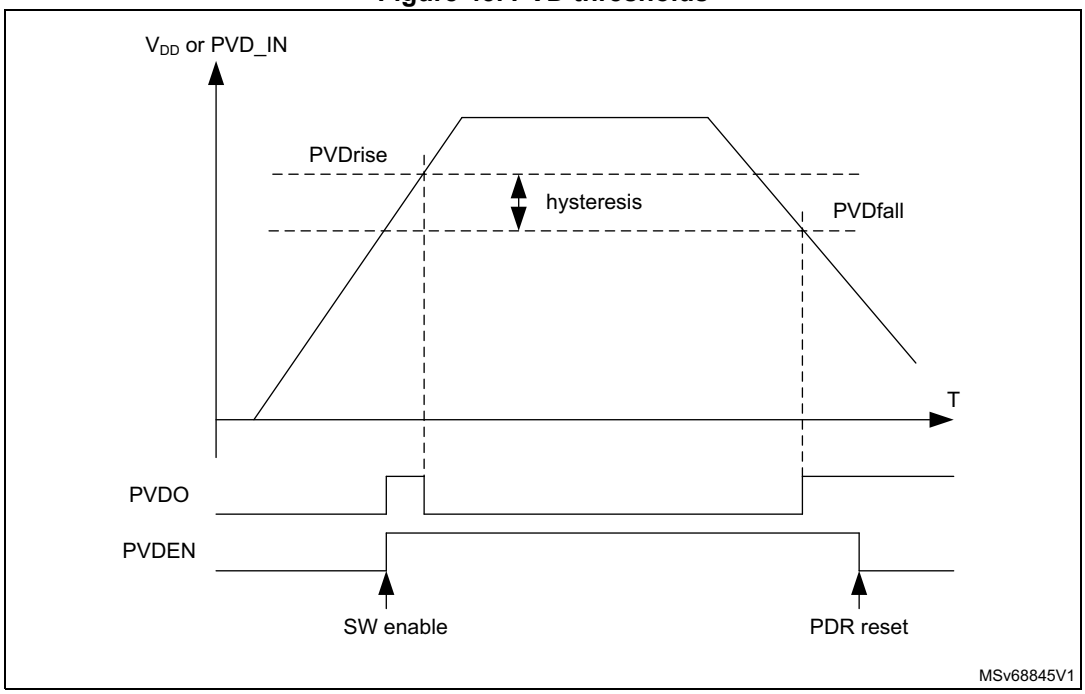
10.6.3 Programmable voltage detector (PVD)

The PVD can be used to monitor the V_{DD} power supply by comparing it to a threshold selected by the PLS[2:0] bits in the *PWR voltage monitor control register (PWR_VMCR)*. The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case PVD_IN voltage is compared to the internal VREFINT level.

The PVD is enabled by setting the PVDE bit in *PWR voltage monitor control register (PWR_VMCR)*.

A PVDO flag is available in the *PWR voltage monitor status register (PWR_VMSR)* to indicate if V_{DD} or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, provided it has been enabled through the EXTI registers. The rising/falling edge sensitivity of the EXTI line must be configured according to PVD output behavior. As an example, if the EXTI line is configured to rising edge sensitivity, the interrupt is generated when V_{DD} or PVD_IN voltage drops below the PVD threshold. The service routine can then start an emergency shutdown.

Figure 45. PVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

10.6.4 Analog voltage detector (AVD)

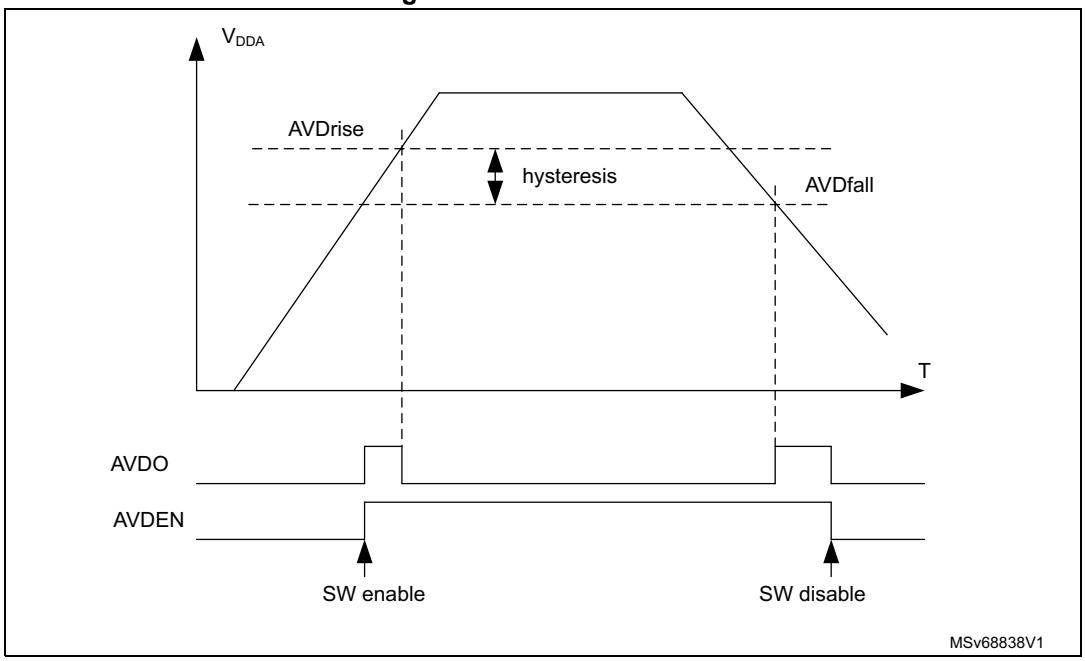
The AVD can be used to monitor the V_{DDA} supply by comparing it to a threshold selected by the ALS[1:0] bits in the *PWR voltage monitor control register (PWR_VMCR)*.

The AVD is enabled by setting the AVDEN bit in *PWR voltage monitor control register (PWR_VMCR)*.

An AVDO flag is available in the *PWR voltage monitor status register (PWR_VMSR)* to indicate whether V_{DDA} is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The AVDO interrupt can be generated when V_{DDA} drops below the AVD threshold and/or

when V_{DDA} rises above the AVD threshold, depending on EXTI rising/falling edge configuration. As an example, the service routine can indicate when the V_{DDA} supply drops below a minimum level.

Figure 46. AVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

10.6.5 V_{DDIO2} voltage monitor (IO2VM)

The IO2VM monitors the independent supply voltage V_{DDIO2} to ensure that the peripheral is in its functional supply range. The VDDIO2RDY flag (see *PWR voltage monitor control register (PWR_VMCR)*) indicates whether a valid supply is present or not.

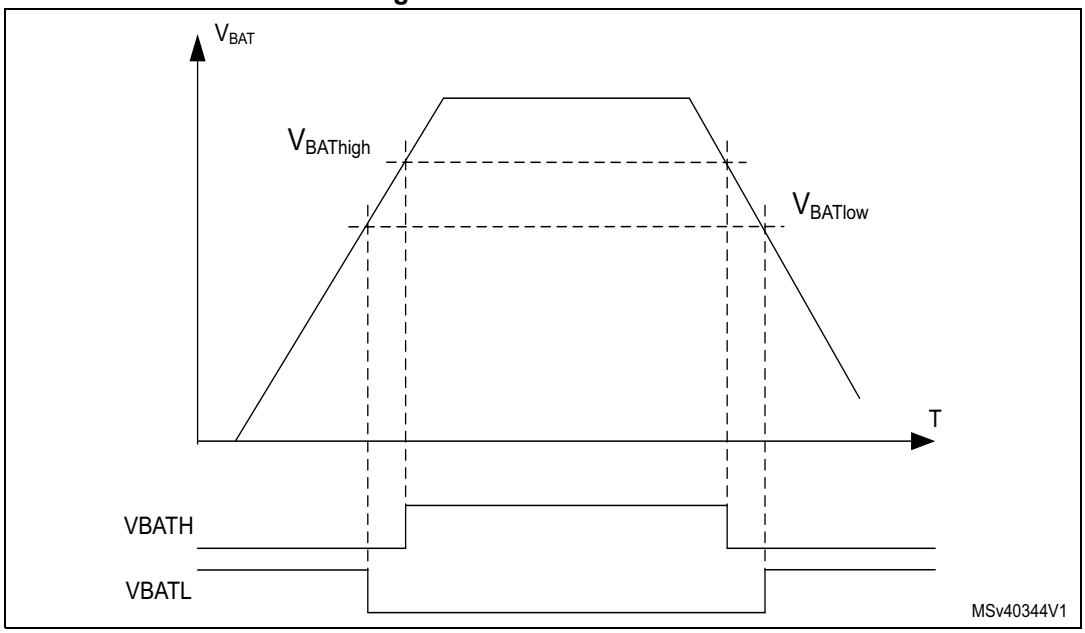
10.6.6 Backup domain voltage monitoring

In VBAT mode, the battery voltage supply (backup domain) can be monitored by comparing it with two threshold levels: $V_{BAThigh}$ and V_{BATlow}. The VBAT supply monitoring can be enabled/disabled via MONEN bit in *PWR Backup domain control register (PWR_BDCR)*. When it is enabled, the battery voltage thresholds increase power consumption.

If the Backup domain voltage monitoring internal tamper is enabled in the TAMP peripheral (ITAMP1E = 1 in the TAMP_CR1 register), a tamper event is generated when the battery voltage is above the functional range.

Note: The Backup domain voltage is V_{DD} when present, V_{BAT} otherwise.

Figure 47. VBAT thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

10.6.7 Temperature monitoring

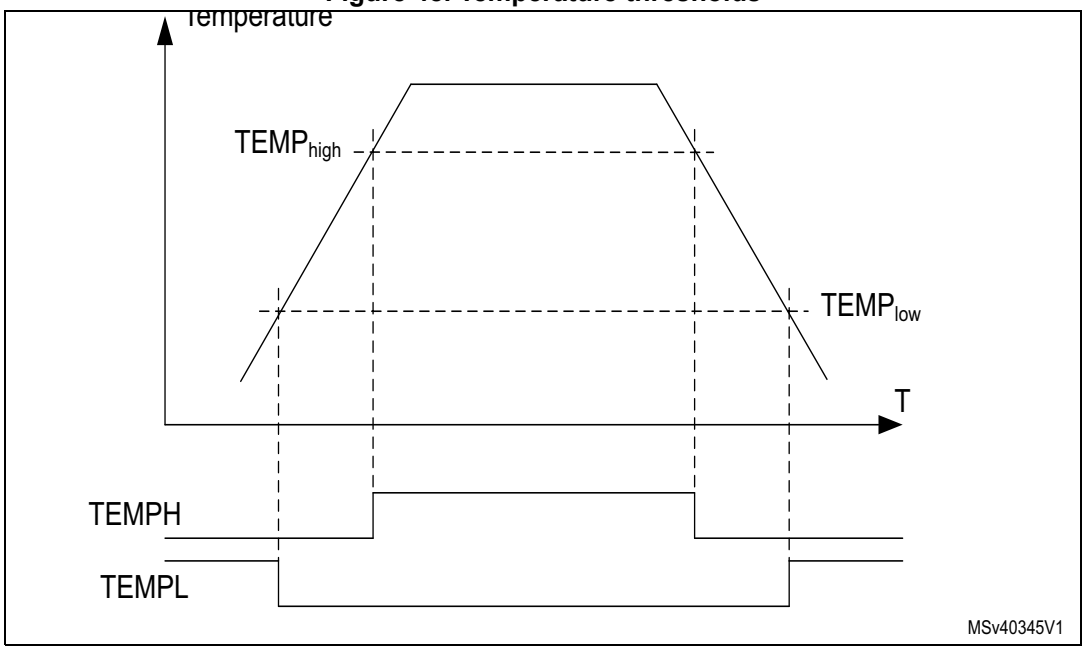
A dedicated temperature sensor cell is embedded in the power control. The junction temperature can be monitored by comparing it with two threshold levels, TEMPhigh and TEMPlow. TEMPH and TEMPL flags in the *PWR Backup domain status register (PWR_BDSR)*, which indicates whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via MONEN bit in *PWR Backup domain control register (PWR_BDCR)*.

When enabled, the temperature thresholds increase power consumption. As an example the levels may be used to trigger a routine to perform temperature control tasks.

If the temperature monitoring internal tamper is enabled in the TAMP peripheral (ITAMP2E = 1 in the TAMP_CR1 register), a tamper event is generated when the temperature is above or below the functional range.

TEMPH and TEMPL wake-up interrupts are available on the RTC tamper signals (see *Section 24: Tamper and backup registers (TAMP)*).

Figure 48. Temperature thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

10.7 PWR management

10.7.1 Voltage scaling

The voltage regulator supporting voltage scaling with the following features:

- Run mode voltage scaling
  - VOS0: scale 0
  - VOS1: scale 1
  - VOS2: scale 2
  - VOS3: scale 3
- Stop mode voltage scaling
  - SVOS3: scale 3
  - SVOS4: scale 4
  - SVOS5: scale 5

For more details on voltage scaling values, refer to the product datasheets.

After reset, the system starts on the lowest Run mode voltage scaling (VOS3). The voltage scaling can be changed on-the-fly by software by programming VOS bits in PWR_VOSCR register, according to the required system performance. When exiting from the Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS3 value.

Before entering Stop mode, the software must preselect the SVOS level in the PWR_PMCR registers. The Stop mode voltage scaling for SVOS4 and SVOS5 also sets the voltage regulator in Low-power mode, to further reduce power consumption.

10.7.2 Power management examples

Figure 49. Dynamic voltage scaling in Run mode

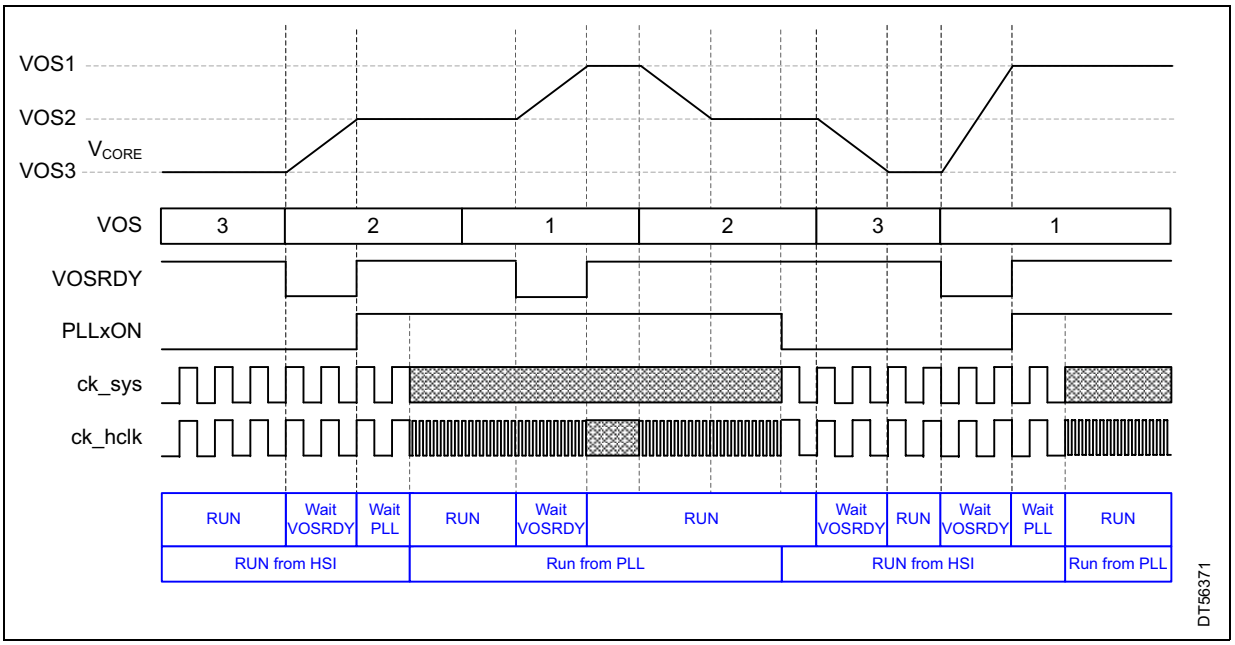


Figure 45 illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS3.
2. The system performance is increased to a medium-speed clock from the PLL with voltage scaling VOS2. To do this:
   a) Program the voltage scaling to VOS2.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The system performance is increased to high-speed clock from the PLL with voltage scaling VOS1. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the VCORE supply has reached the required level indicated by VOSRDY, increase the clock frequency.
4. The system performance is reduced to a medium-speed clock with voltage scaling VOS2. To do this:
   a) First decrease the system frequency.
   b) Then decrease the voltage scaling to VOS2.
5. The next step is to reduce the system performance to the HSI clock with voltage scaling VOS3. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS3.

6. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS1.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS must be set accordingly, otherwise the system can be unreliable.

10.8 Power modes

By default, the microcontroller is in Run mode after a system or a power reset. Several low-power modes are available to reduce consumption when there is no need to keep the CPU running, for example when waiting for an external event. The user can select the mode that gives the best compromise between low-power consumption, short startup time, and wake-up sources.

The device features the following low-power modes:

- Sleep mode

  CPU clock off, peripherals such as NVIC and SysTick can run and wake-up the CPU when an interrupt or an event occurs. Refer to *Section 10.8.4*.

- Stop mode

  Achieves the lowest power consumption, while retaining the content of SRAM and registers. All clocks in the core domain are stopped. The PLL, the HSE crystal oscillators, HSI (except if HSIKERON is set), HSI48 and CSI RC (except if CSIKERON is set) are disabled. The LSE or LSI is still running.

  The RTC can remain active (Stop mode with RTC, Stop mode without RTC).

  The system clock when exiting from Stop mode can be either HSI up to 64 MHz or CSI, depending on software configuration.

  Refer to *Section 10.8.5*.

- Standby mode

  This mode achieves the lowest power consumption with BOR. The internal regulator is switched off so that the core domain is powered off. The PLL, the HSI RC, HSI48, the CSI RC, and the HSE crystal oscillators are also switched off.

  The RTC can remain active (Standby mode with RTC, Standby mode without RTC).

  The brownout reset (BOR) remains active.

  The state of the I/O (except I/Os used by standby mode) during Standby mode can be retained.

  After entering Standby mode, SRAMs and register contents are lost except for registers and backup SRAM in the Backup domain and Standby circuitry.

  The device exits Standby mode when an external reset (NRST pin), an IWDG reset, WKUP pin event (configurable rising or falling edge), an RTC event occurs (alarm, periodic wake-up, timestamp), or a tamper detection. The tamper detection can be

raised either due to external pins or due to an internal failure detection.

The system clock after wake-up is HSI at 32 MHz.

Refer to *Section 10.8.6*.

Table 97 shows the power modes overview.

Table 101. Low-power mode summary

Mode name	Entry	Wake-up source(1)	Wake-up system clock	Effect on clocks	Voltage regulators
Sleep (Sleep-now or Sleep-on-exit)	WFI or Return from ISR	Any interrupt	Same as before entering Sleep mode	– CPU clock OFF – No effect on other clocks or analog clock sources	VOS3, VOS2, VOS1, or VOS0
	WFE	Wake-up event			
Stop	LPMS = 0 + SLEEPDEEP bit + WFI or Return from ISR or WFE	– Any EXTI line (configured in the EXTI registers) – Specific peripherals events(2)	– CSI when STOPWUCK = 1 in RCC_CFGR – HSI with the frequency before entering Stop mode, up to 64 MHz, when STOPWUCK = 0	– All clocks OFF except LSI and LSE – HSI or CSI can be enabled temporarily when requested by software	SVOS3, SVOS4, or SVOS5
Standby	LPMS = 1 + SLEEPDEEP bit + WFI or Return from ISR or WFE	WKUP pin edge, RTC event, IWDG reset, external reset in NRST pin	HSI clock at 64 MHz	All clocks OFF except LSI and LSE	OFF

1. Refer to *Table 98*.
2. Peripherals able to wake-up the system from Stop mode (possible only when SVOS3 is selected before entering Stop mode).

Table 102. Functionalities depending on the working mode(1)

Peripheral	Run	Sleep	Stop				Standby		VBAT
			Available	Wake-up capability			Available	Wake-up capability	
				SVOS3	SVOS4	SVOS5			
CPU	Y	-	-	-	-	-	-	-	-
Flash memory	O	O	(2)	-	-	-	-	-	-
SRAM1	Y(3)	Y(3)	O(4)	-	-	-	-	-	-
SRAM2	Y(3)	Y(3)	O(4)	-	-	-	-	-	-
SRAM3	Y(3)	Y(3)	O(4)	-	-	-	-	-	-
BKPSRAM	O	O	O	-	-	-	O	-	O
FMC	O	O	-	-	-	-	-	-	-
OCTOSPI1	O	O	-	-	-	-	-	-	-
Backup registers	Y	Y	Y	-	-	-	Y	-	Y
Brownout reset (BOR)	Y	Y	Y	-	-	-	Y	-	-

Table 102. Functionalities depending on the working mode(1) (continued)

Peripheral	Run	Sleep	Stop				Standby		VBAT
			Available	Wake-up capability			Available	Wake-up capability	
				SVOS3	SVOS4	SVOS5			
Programmable voltage detector (PVD)	O	O	O	O	O	O	-	-	-
Analog voltage detector (AVD)	O	O	O	O	O	O	-	-	-
GPDMA	O	O	-	-	-	-	-	-	-
High-speed internal (HSI)	O	O	-	-	-	-	-	-	-
Oscillator HSI48	O	O	-	-	-	-	-	-	-
High-speed external (HSE)	O	O	-	-	-	-	-	-	-
Low-speed internal (LSI)	O	O	O	O	-	-	O	-	-
Low-speed external (LSE)	O	O	O	-	-	-	O	-	O
Low-power RC oscillator (CSI)	O	O	-	-	-	-	-	-	-
Clock security system (CSS)	O	O	-	-	-	-	-	-	-
Clock security system on LSE	O	O	O	O	O(5)	O(5)	O	O	-
Backup domain voltage and temperature monitoring	O	O	O	O	O(5)	O(5)	O	O	O
RTC/TAMP	O	O	O	O	O	O	O	O	O
Number of TAMP tamper pins	8	8	8	-	-	-	4	-	2
USB FS, UCPD	O	O	O	O	-	-	-	-	-
USARTx	O	O	O	O	-	-	-	-	-
Low-power UART (LPUART)	O	O	O	O	-	-	-	-	-
I2Cx	O	O	O	O	-	-	-	-	-
I3Cx	O	O	O	O	-	-	-	-	-
HDMI_CEC	O	O	O	O	-	-	-	-	-
SPIx	O	O	O	O	-	-	-	-	-
FDCANx	O	O	-	-	-	-	-	-	-
SDMMCx	O	O	-	-	-	-	-	-	-
Ethernet	O	O	O	O	-	-	-	-	-
SAIx	O	O	-	-	-	-	-	-	-
ADCx (x = 1,2)	O	O	-	-	-	-	-	-	-
DAC1 (2 converters)	O	O	O	-	-	-	-	-	-
VREFBUF	O	O	O	-	-	-	-	-	-
Temperature sensor (DTS)	O	O	O	O	-	-	-	-	-
Timers (TIMx)	O	O	-	-	-	-	-	-	-
Low-power timer LPTIMx	O	O	O	O	-	-	-	-	-

Table 102. Functionalities depending on the working mode[1] (continued)

Peripheral	Run	Sleep	Stop				Standby		VBAT
			Available	Wake-up capability			Available	Wake-up capability	
				SVOS3	SVOS4	SVOS5			
Independent watchdog (IWDG)	O	O	O	O	O[5]	O[5]	O	O	-
Window watchdog (WWDG)	O	O	-	-	-	-	-	-	-
SysTick timer (SYSTICK)	O	O	O	-	-	-	-	-	-
Digital camera interface (DCMI)	O	O	-	-	-	-	-	-	-
Parallel synchronous slave interface (PSSI)	O	O	-	-	-	-	-	-	-
CORDIC coprocessor (CORDIC)	O	O	-	-	-	-	-	-	-
Filter mathematical accelerator (FMAC)	O	O	-	-	-	-	-	-	-
Random number generator (RNG)	O	O	-	-	-	-	-	-	-
AES and secure AES (AES, SAES)	O	O	-	-	-	-	-	-	-
Public key accelerator (PKA)	O	O	-	-	-	-	-	-	-
On-the-fly decryption (OTFDEC)	O	O	-	-	-	-	-	-	-
HASH accelerator	O	O	-	-	-	-	-	-	-
CRC calculation unit	O	O	-	-	-	-	-	-	-
GPIOs	O	O	-	-	-	-	O[6]	O[7]	-
EXTI	O	O	O	O	O	O	-	-	-

1. Y = yes (enabled). O = optional (disabled by default, can be enabled by software). - = not available. HSI or CSI are available as kernel clock for peripherals only in SVOS3.
2. The memory can be configured in Low-power mode. By default, it is not in Low-power mode during Stop (SVOS3, SVOS4).
3. The SRAM clock can be gated on or off independently. By default clock is enabled in Run and Sleep modes.
4. The SRAMs can be powered on or off independently. By default, they are not in Power-off mode during Stop.
5. Wake-up with internal tamper.
6. GPIOs state can be retained during Standby mode. By default GPIOs states are not retained.
7. 8 pins are capable of wake-up from Standby mode: PA0, PA2, PB7, PC1, PC13, PD2, PD3 and PI8 (not available on all devices, check the datasheet).

In addition, the power consumption in Run mode can be reduced by slowing down the system clocks, configuring voltage scaling to lower power ranges, and by gating the clocks to the APB and AHB peripherals when they are not used.

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while the debug features are used. This is due to the fact that the Cortex-M33 core is no longer clocked.

However, by setting some configuration bits in the DBGMCU control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Section 75.2.5: Debug and low-power modes*.

10.8.1 Slowing down system clocks

In Run mode, the speed of the system clocks (SYSCLK, HCLK, PCLK) can be reduced by programming the prescaler registers. These prescalers can also be used to slow down the peripherals before entering the Sleep mode.

For more details, refer to *Section 11: Reset and clock control (RCC)*.

10.8.2 Peripheral clock gating

In Run mode, the HCLK and PCLK for individual peripherals and memories can be stopped at any time to reduce the power consumption.

To further reduce the power consumption in Sleep mode, the peripheral clocks can be disabled before executing the WFI or WFE instructions.

The peripheral clock gating is controlled by the RCC_AHBxENR and RCC_APBxENR registers.

Disabling the peripherals clocks in Sleep mode can be performed automatically by resetting the corresponding bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers.

10.8.3 Low-power modes

Entering into a low-power mode

The MCU enters in low-power modes by executing the WFI (wait for interrupt), or WFE (wait for event) instructions, or when the SLEEPONEXIT bit in the Cortex-M33 system control register is set on Return from ISR.

Entering into a low-power mode through WFI or WFE is executed only if no interrupt is pending or no event is pending.

Exiting a low-power mode

The MCU exits the Sleep or Stop mode according to how the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the device.
- If the WFE instruction is used to enter the low-power mode, the MCU exits the low-power mode as soon as an event occurs. The wake-up event can be generated either by:
  - an NVIC IRQ interrupt:
    - When SEVONPEND = 0 in the Cortex-M33 system control register
      By enabling an interrupt in the peripheral control register and in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and. the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. Only NVIC interrupts with high enough priority can wake up and interrupt the MCU.
    - When SEVONPEND = 1 in the Cortex-M33 system control register
      By enabling an interrupt in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and when enabled the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) must be cleared. All NVIC interrupts

wake up the MCU, even the disabled ones. Only enabled NVIC interrupts with high enough priority can wake up and interrupt the MCU.

– an event:
  > Configuring an EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit, as the pending bits corresponding to the event line are not set. It may be necessary to clear the interrupt flag in the peripheral.

The MCU exits Standby mode through an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC/TAMP event (see *Figure 624: RTC block diagram*).

After waking up from Standby mode, the program execution restarts in the same way as after a reset (boot pin sampling, option bytes loading, reset vector is fetched).

Caution: When the device is in Stop mode, a peripheral interrupt powers on an internal oscillator. The corresponding NVIC interrupt channel must be enabled to allow the interrupt to exit the device from Stop mode. It is not allowed to disable a peripheral interrupt by disabling only the NVIC channel while keeping the peripheral interrupt enable, as the device could remain in Stop mode with clock ON.

10.8.4 Sleep mode

I/O states in Sleep mode

In Sleep mode, all I/O pins keep the same state as in Run mode.

Entering the Sleep mode

The MCU enters the Sleep mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is clear (see the table below for details on how to enter the Sleep mode).

Exiting the Sleep mode

The MCU exits the Sleep mode as described in *Exiting a low-power mode* (see the table below for details on how to exit the Sleep mode).

Table 103. Sleep mode

Sleep mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP = 0 – No interrupt (for WFI) or event (for WFE) pending Refer to the Cortex-M33 system control register.
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 – No interrupt pending Refer to the Cortex-M33 system control register.

Table 103. Sleep mode (continued)

Sleep mode	Description
Mode exit	If WFI or Return from ISR was used for entry Interrupt (see *Table 146: STM32H562/563/573xx vector table*) If WFE was used for entry and SEVONPEND = 0: Wake-up event (see *Section 18.3: EXTI functional description*) If WFE was used for entry and SEVONPEND = 1: Interrupt even when disabled in NVIC (see *Table 146: STM32H562/563/573xx vector table*) or wake-up event (see *Section 18.3: EXTI functional description*)
Wake-up latency	None

10.8.5 Stop mode

The Stop mode is based on the Cortex-M33 Deepsleep mode combined with the peripheral clock gating. The voltage regulator is configured by SVOSx bits (the selected SVOS4 and SVOS5 levels add an additional startup delay when exiting from system Stop mode). In Stop mode, all clocks in the core domain are stopped. The PLL, HSI, HSI48, CSI and HSE oscillators are disabled.

It is possible to keep the HSI or CSI clock enabled during Stop mode, to be quickly available as kernel clock for peripherals.

All SRAMs and register contents are preserved, but the SRAMs can be totally or partially switched off to further reduced consumption. The user can select which memory is discarded during Stop mode by means of xxSO bits in PWR_PMCR registers.

Table 104. Memory shut-off block selection

Selection bit	Shut-off block in Stop mode	STM32H523/533xx	STM32H562/563/573xx
SRAM1SO	AHB SRAM1	X	X
SRAM2_48SO	AHB SRAM2 48-Kbyte	X	X
SRAM2_16SO	AHB SRAM2 16-Kbyte	-	X
SRAM2_H16SO	AHB SRAM2 16-Kbyte	X	-
SRAM2_L16SO	AHB SRAM2 16-Kbyte	X	-
SRAM3SO	AHB SRAM3	X	
ETHERNETSO	ETHERNET RAM	-	X

The BOR is always available in Stop mode.

I/O states in Stop mode

In the Stop mode, all I/O pins keep the same state as in the Run mode.

Entering the Stop mode

The MCU enters the Stop mode as described in *Entering into a low-power mode*, when the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 101* for details on how to enter the Stop mode).

If the flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, the Stop mode entry is delayed until the APB access is finished.

In Stop mode, the following features can be selected by programming the individual control bits:

- The independent watchdog (IWDG) is started by writing to its key register or by hardware option. Once started, it can be stopped only by a reset (see *Section 44.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in the *RCC backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock is configured by the LSION bit in RCC_BCDR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BCDR.

The AVD and the PVD can be used in Stop mode. If they are not needed, they must be disabled by software to save their power consumptions.

The ADCx (x = 1, 2), the DAC1 (two channels), the temperature sensor, and the VREFBUF can consume power during the Stop mode, unless they are disabled before entering this mode.

Exiting the Stop mode

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered. Some peripherals are able to wake up the system (refer to *Table 138: EXTI line connections*) from Stop mode, this is only possible when SVOS3 is selected before entering Stop mode.

Note: When wake-up from Stop with peripherals is needed, SVOS3 must be selected.

When exiting Stop mode by issuing an interrupt or a wake-up event, CSI is selected as system clock if bit STOPWUCK is set in RCC clock configuration register 1 (RCC_CFGR). The HSI oscillator is selected as system clock if STOPWUCK is cleared. The wake-up time is shorter when CSI is selected as wake-up system clock. The HSI selection allows a wake-up at higher frequency (up to 64 MHz).

The MCU exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered.

When exiting the Stop mode, the MCU is in Run mode, VOS3.

Table 105. Stop mode

Stop mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 000 in PWR_CR1
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 0 in PWR_PMCR
	Note: To enter Stop mode, all EXTI line pending bits (in the EXTI rising edge pending register 2 (EXTI_RPR2)), and the peripheral flags generating wake-up interrupts must be cleared. Otherwise, the Stop mode entry procedure is ignored and the program execution continues.
Mode exit	If WFI or Return from ISR was used for entry: - any EXTI line configured in interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 146: STM32H562/563/573xx vector table*). - any peripheral interrupt occurring when the AHB/APB clocks are present due to an autonomous peripheral clock request (the peripheral vector must be enabled in the NVIC) If WFE was used for entry and SEVONPEND = 0: - any EXTI line configured in event mode (see *Section 18.3: EXTI functional description*). If WFE was used for entry and SEVONPEND = 1: - any EXTI line configured in interrupt mode (even if the corresponding EXTI interrupt vector is disabled in the NVIC). The interrupt source can be external interrupts or peripherals with wake-up capability (see *Table 146: STM32H562/563/573xx vector table*). - any EXTI line is configured in event mode (see *Section 18.3: EXTI functional description*)
	Note: All peripheral clocks must be enabled to allow this peripheral to generate a wake-up from Stop interrupt ([PERIPH]EN and [PERIPH]LPEN bits must be set in the RCC, and a functional independent clock must be selected).
Wake-up latency	Longest wake-up time between: HSI or CSI wake-up time and flash memory wake-up time from Stop mode.

10.8.6 Standby mode

The lowest power mode in which the BOR is active is the Standby mode. It is based on the Cortex-M33 Deepsleep mode, with the voltage regulators disabled. The PLL, HSI, HSI48, CSI, and HSE oscillators are also switched off.

The SRAMs and register contents are lost except for registers in the Backup domain and Standby circuitry (see *Figure 36*).

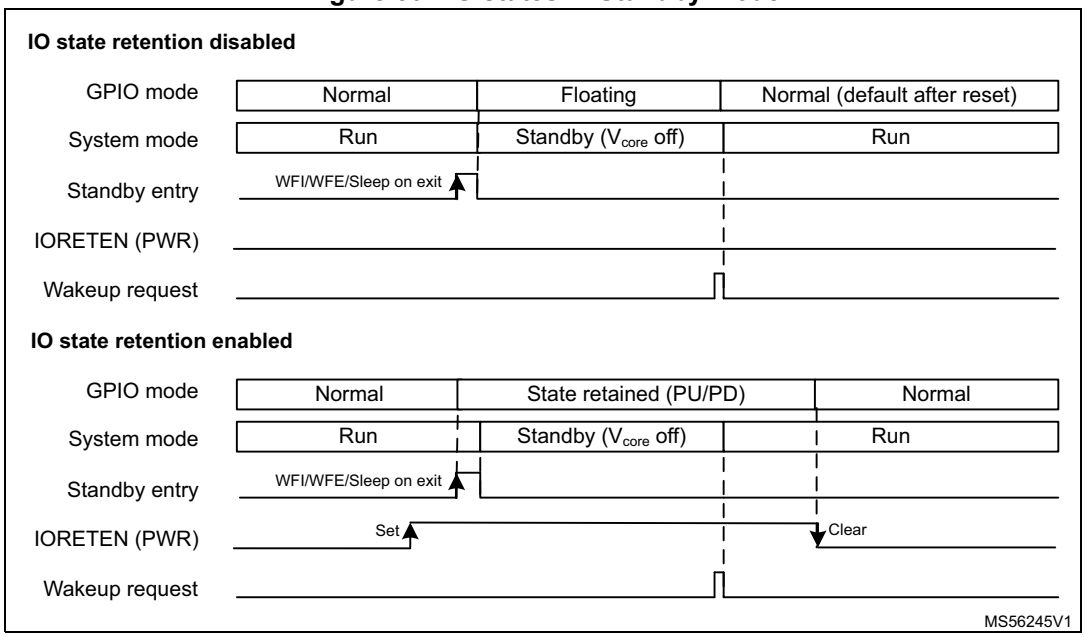
The BOR is always available in Standby mode.

I/O states in Standby mode

In the Standby mode, the I/Os are by default in floating state. If the IORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. I/O retention mode is enabled for all I/Os except those supporting the standby functionality and JTAG I/Os (PA13, PA14, PA15, and PB4). When entering into Standby mode, the state of the output is sampled, and pull-up or pull-down resistor are set to maintain the I/O output during Standby mode.

If the JTAGIORETEN bit in the PWR_IORETR register is set, the I/Os output state is retained. I/O retention mode is enabled for PA13, PA14, PA15, and PB4 (default JTAG pull-up/pull-down after wake-up is not enabled).

Figure 50. I/O states in Standby mode



After wake-up from Standby mode, as long as IORETEN (or JTAGIORETEN for JTAG I/Os) is set, the retained stated (pull-up/pull-down) remains applied.

The GPIO pin state before standby can be identified with GPIO_IDR register (when both GPIO port clock and input buffer are enabled).

The application can release the I/O state (clear the retained Pull-up/Pull-down) by clearing the IORETEN (or JTAGIORETEN for JTAG IOs) bit, before or after reconfiguring the GPIOs and related peripherals. The GPIOs can then be configured in a known state before releasing the retained state.

The RTC outputs on PC13 and PI8[(a)] are functional in Standby mode. PC14 and PC15 used for LSE are also functional. Eight wake-up (WKUPx, x = 1 to 8) and four RTC tamper pins are available.

Entering Standby mode

The MCU enters the Standby mode as described in *Entering into a low-power mode*, when

a. Not available on all devices, check the datasheet.

the SLEEPDEEP bit in the Cortex-M33 system control register is set (see *Table 102* for details on how to enter Standby mode).

In Standby mode, the following features can be selected by programming individual control bits:

- The independent watchdog (IWDG) is started by writing to its Key register or by hardware option. Once started, it can be stopped only by a reset (see *Section 44.4: IWDG functional description*).
- The real-time clock (RTC) is configured by the RTCEN bit in *RCC backup domain control register (RCC_BDCR)*.
- The internal RC oscillator LSI clock, is configured by the LSION bit in RCC_BDCR.
- The external 32.768 kHz oscillator (LSE) is configured by the LSEON bit in RCC_BDCR.
- The I/Os retention is configured by the IORETEN bit in the PWR_IORETR register.

Exiting Standby mode

The MCU exits the Standby mode as described in *Exiting a low-power mode*. The SBF status flag in the *PWR status register (PWR_PMSR)* indicates that the MCU was in Standby mode. All registers are reset after wake-up from Standby except for *PWR Backup domain control register (PWR_BDCR)* and *PWR I/O retention register (PWR_IORETR)* (see *Table 106* for more details on how to exit Standby mode).

When exiting Standby mode, I/Os output state that were retained during Standby through IORETEN bit, keep this configuration upon exiting Standby mode until the IORETEN bit in PWR_IORETR register is cleared. Once IORETEN is cleared, the I/Os are configured to their reset values, or to the pull-up/pull-down state according to the GPIOx_PUPDR registers.

For I/Os, with a pull-up or pull-down predefined after reset (some JTAG/SWD I/Os), in case those pull-up or pull-down are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Also in case the GPIOx_PUPDR values programed after exiting from Standby are different from the retained values during Standby, both a pull-down and pull-up are applied until IORETEN is cleared, releasing the retained value.

Table 106. Standby mode

Standby mode	Description
Mode entry	WFI (wait for interrupt) or WFE (wait for event) while: – SLEEPDEEP bit is set in Cortex-M33 system control register – No interrupt (for WFI) or event (for WFE) pending – LPMS = 1 in PWR_PMCR – WUFx bits cleared in PWR_WUSR
	On Return from ISR while: – SLEEPDEEP bit is set in Cortex-M33 system control register – SLEEPONEXIT = 1 – No interrupt pending – LPMS = 1 in PWR_PMCR – WUFx bits cleared in PWR_WUSR – RTC/TAMP flags corresponding to the chosen wake-up source, cleared
Mode exit	WKUPx pin edge, RTC event, external Reset in NRST pin, IWDG Reset, BOR reset
Wake-up latency	Reset phase

10.8.7 Power modes output pins

In order to help the debug, three signals are available as device pins alternate functions:

- **CSLEEP**

  When set, CSLEEP indicates that the system is in Sleep mode: WFI or WFE has been executed.

  When cleared, CSLEEP indicates that the system is in Run mode.

- **CDSTOP**

  When set, CDSTOP indicates that the system is in CStop mode, meaning that the following conditions are fulfilled:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  – No AHB/APB clock is running.

  When cleared, CDSTOP indicates that the system is not in CStop mode: AHB/APB clocks are running.

The table below explains the MCU power mode depending on these signals states.

Table 107. Power modes output states versus MCU power modes

CSLEEP	CSTOP	MCU power modes(1)
0	0	Run mode
1	0	Sleep mode or Stop mode, with AHB/APB clocks running
1	1	Stop mode

1. CSLEEP and CDSTOP are generated in core domain, consequently they are not driven in Standby mode.

10.9 PWR security and privileged protection

10.9.1 PWR security protection

When the TrustZone security is activated by the TZEN option byte in the flash memory option byte configuration register, some PWR register fields can be secured against non-secure access.

The PWR TrustZone security allows the following features to be secured through the PWR_SECCFGR register:

- Low-power mode
- Wake-up (WKUP) pins
- Voltage detection and monitoring
- VBAT mode
- I/Os retention configuration

Other PWR configuration bits are secure when:

- The system clock selection is secure in RCC: the voltage scaling (VOS) configuration is secure.
- The UCPD1 is secure in the GTZC: the PWR_UCPDR register is secure.

Table 108 gives a summary of the PWR secured bits following the security configuration bit in PWR_SECCFGR.

A non-secure access to a secure-protected register bit is denied:

- The secured bits are not written (WI) with a non-secure write access.
- The secured bits are read as 0 (RAZ) with a non-secure read access.

A non-secure write access to PWR_SECCFGR is WI and generates an illegal access event and an interrupt if enabled in the GTZC. It can be read with a non-secure read access.

When the TrustZone security is disabled (TZEN = 0xC3), PWR_SECCFGR is RAZ/WI and all other registers are non-secure.

Table 108. PWR security configuration summary

Secure configuration register	Security configuration bit	Register name	Secured bits	Non-secure access on secure bits
PWR_SECCFGR	Not applicable(1)	PWR_SECCFGR	All bits	Read OK. WI and illegal access event
PWR_SECCFGR	At least one bit is set	PWR_PRIVCFGR	SPRIV	Read OK. WI
PWR_SECCFGR	LPMSEC	PWR_PMCR	All bits	WI
PWR_SECCFGR	VUSBSEC	PWR_USBSCR	All bits	WI
PWR_SECCFGR	VBSEC	PWR_BDCR	All bits	WI
		PWR_DBPCR	All bits	WI
PWR_SECCFGR	RETSEC	PWR_IORETR	All bits	WI

Table 108. PWR security configuration summary (continued)

Secure configuration register	Security configuration bit	Register name	Secured bits	Non-secure access on secure bits
PWR_SECCFGR	WUPxSEC (x = 1 to 8)	PWR_WUCR	WUPENx	RAZ/WI
			WUPPx	RAZ/WI
			WUPPUPD	RAZ/WI
		PWR_WUSCR	CWUFx	WI
GTZC_TZSC_SECCFGR	UCPD1SEC	PWR_UCPDR	All bits	RAZ/WI
RCC_SECCFGR	SYSCLKSEC	PWR_VOSCR	VOS[1:0]	RAZ/WI
RCC_SECCFGR	SCMSEC	PWR_SCCR	All bits	WI
RCC_SECCFGR	SCMSEC	PWR_VMCR	All bits	WI

1. PWR_SECCFGR is always secure.

10.9.2 PWR privileged protection

By default, after a reset, all registers can be read or written with both privileged and unprivileged accesses, except PWR_PRIVCFGR, which can be written only with privileged access. PWR_PRIVCFGR can be read by secure and non secure, privileged and unprivileged accesses.

The SPRIV bit in PWR_PRIVCFGR can be written only with secure privileged access. This bit configures the privileged access of all PWR secure functions (defined by PWR_SECCFGR, GTZC, RCC, or GPIO, as shown in *Table 108*).

When the SPRIV bit is set in PWR_PRIVCFGR:

- The PWR secure bits can be written only with privileged access, including PWR_SECCFGR.
- The PWR secure bits can be read only with privileged access except PWR_SECCFGR and PWR_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as 0 and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit of PWR_PRIVCFGR can be written only with privileged access, secure or non-secure. It configures the privileged access of all PWR securable functions configured as non-secure (defined by PWR_SECCFGR, GTZC, RCC, or GPIO, see *Table 104*).

When the NSPRIV bit is set in PWR_PRIVCFGR:

- The PWR securable bits configured as non-secure can be written only with privileged access.
- The PWR securable bits configured as non-secure can be read only with privileged access except PWR_PRIVCFGR, which can be read by privileged or unprivileged accesses.
- The VOSRDY and BOOSTRDY bits in PWR_VOSR, PWR_SR, PWR_SVMSR, PWR_BDSR and PWR_WUSR can be read with privileged or unprivileged accesses.
- An unprivileged access to a privileged PWR bit or register is discarded: the bits are read as 0, and the write to these bits is ignored (RAZ/WI).

10.10 PWR interrupts

Table 109 gives a summary of the interrupt sources and the way to control them.

Table 109. PWR interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep, Stop modes	Exit Standby modes
PVD/AVD output	Programmable voltage detector through EXTI line 16	PVDO/AVDO	EXTI line 16 enabled	Write EXTI PIF16 = 1	Yes	No

10.11 PWR registers

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

10.11.1 PWR power mode control register (PWR_PMCR)

STM32H562/572/573xx devices only.

This register is protected against non-secure access when LPMSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when LPMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when LPMSEC = 0 and NSPRIV = 1.

Address offset: 0x000

Reset value: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SRAM1SO	SRAM2_48SO	SRAM2_16SO	SRAM3SO	Res.	Res.	Res.	Res.	Res.	Res.	ETHERNETSO
					rw	rw	rw	rw							rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AVD_READY	BOOSTE	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	SVOS[1:0]		Res.	LPMS
		rw	rw			rw		rw				rw	rw		rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SRAM1SO**: AHB SRAM1 shut-off in Stop mode
0: AHB RAM1 content is kept in Stop mode.
1: AHB RAM1 content is lost in Stop mode.

Bit 25 **SRAM2_48SO**: AHB SRAM2 48-Kbyte shut-off in Stop mode.
0: AHB RAM2 48-Kbyte content is kept in Stop mode.
1: AHB RAM2 48-Kbyte content is lost in Stop mode.

Bit 24 **SRAM2_16SO**: AHB SRAM2 16-Kbyte shut-off in Stop mode.
0: AHB RAM2 16-Kbyte content is kept in Stop mode.
1: AHB RAM2 16-Kbyte content is lost in Stop mode.

Bit 23 **SRAM3SO**: AHB SRAM3 shut-off in Stop mode.
0: AHB RAM3 content is kept in Stop mode.
1: AHB RAM3 content is lost in Stop mode.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **ETHERNETSO**: ETHERNET RAM shut-off in Stop mode.
0: ETHERNET RAM content is kept in Stop mode.
1: ETHERNET RAM content is lost in Stop mode.

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

Bit 13 **AVD_READY**: analog voltage ready

This bit is used only when the analog switch boost must be enabled (see BOOSTE bit).
It must be set by software when the expected V_{DDA} analog supply level is available.
The correct analog supply level is indicated by the AVDO bit (PWR_VMSR register) after setting the AVDEN bit (PWR_VMCR register) and selecting the supply level to be monitored (ALS bits).
0: peripheral analog voltage V_{DDA} not ready (default)
1: peripheral analog voltage V_{DDA} ready.

Bit 12 **BOOSTE**: analog switch V_{BOOST} control

This bit enables the booster to guarantee the analog switch AC performance when the V_{DD} supply voltage is below 2.7 V (reduction of the total harmonic distortion to have the same switch performance over the full supply voltage range) The V_{DD} supply voltage can be monitored through the PVD and the PLS bits.
0: booster disabled (default)
1: booster enabled if analog voltage ready (AVD_READY = 1)

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **FLPS**: flash memory low-power mode in Stop mode

This bit is used to obtain the best trade-off between low-power consumption and restart time when exiting from Stop mode.
When it is set, the flash memory enters low-power mode when the system is in Stop mode.
0: flash memory remains in normal mode when the system enters Stop mode (quick restart time).
1: flash memory enters low-power mode when the system enters Stop mode (low-power consumption).

Note: When system enters Stop mode with SVOS5 enabled, flash memory is automatically forced in low-power mode.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **CSSF**: clear Standby and Stop flags (always read as 0)

This bit is cleared to 0 by hardware.
0: no effect
1: STOPF and SBF flags cleared

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:2 **SVOS[1:0]**: system Stop mode voltage scaling selection

These bits control the V_{CORE} voltage level in system Stop mode, to obtain the best trade-off between power consumption and performance.
00: reserved
01: SVOS5 scale 5
10: SVOS4 scale 4
11: SVOS3 scale 3 (default)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **LPMS**: low-power mode selection

This bit defines the Deepsleep mode.
0: keeps Stop mode when entering DeepSleep.
1: allows Standby mode when entering DeepSleep.

10.11.2 PWR power mode control register [alternate] (PWR_PMCR)

STM32H523/533xx devices only.

This register is protected against non-secure access when LPMSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when LPMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when LPMSEC = 0 and NSPRIV = 1.

Address offset: 0x000

Reset value: 0x0000 000C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	SRAM1SO	SRAM2_48SO	SRAM2_16HSO	SRAM2_16LSO	SRAM3SO	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AVD_READY	BOOSTE	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	SVOS[1:0]		Res.	LPMS
		rw	rw			rw		rw				rw	rw		rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **SRAM1SO**: AHB SRAM1 shut-off in Stop mode
0: AHB RA.M1 content is kept in Stop mode.
1: AHB RAM1 content is lost in Stop mode.

Bit 26 **SRAM2_48SO**: AHB SRAM2 48-Kbyte shut-off in Stop mode.
0: AHB RAM2 48-Kbyte content is kept in Stop mode.
1: AHB RAM2 48-Kbyte content is lost in Stop mode.

Bit 25 **SRAM2_16HSO**: AHB SRAM2 high 16-Kbyte shut-off in Stop mode.
0: AHB RAM2 high 16-Kbyte content is kept in Stop mode.
1: AHB RAM2]high16-Kbyte content is lost in Stop mode.

Bit 24 **SRAM2_16LSO**: AHB SRAM2 low 16-Kbyte shut-off in Stop mode.
0: AHB RAM2 low 16-Kbyte content is kept in Stop mode.
1: AHB RAM2 low 16-Kbyte content is lost in Stop mode.

Bit 23 **SRAM3SO**: AHB SRAM3 shut-off in Stop mode.
0: AHB RAM3 content is kept in Stop mode.
1: AHB RAM3 content is lost in Stop mode.

Bits 22:14 Reserved, must be kept at reset value.

Bit 13 **AVD_READY**: analog voltage ready
This bit is used only when the analog switch boost must be enabled (see BOOSTE bit).
It must be set by software when the expected V_{DDA} analog supply level is available.
The correct analog supply level is indicated by the AVDO bit (PWR_VMSR register) after setting the AVDEN bit (PWR_VMCR register) and selecting the supply level to be monitored (ALS bits).
0: peripheral analog voltage V_{DDA} not ready (default)
1: peripheral analog voltage V_{DDA} ready.

Bit 12 **BOOSTE**: analog switch V_{BOOST} control

This bit enables the booster to guarantee the analog switch AC performance when the V_{DD} supply voltage is below 2.7 V (reduction of the total harmonic distortion to have the same switch performance over the full supply voltage range) The V_{DD} supply voltage can be monitored through the PVD and the PLS bits.

0: booster disabled (default)

1: booster enabled if analog voltage ready (AVD_READY = 1)

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **FLPS**: flash memory low-power mode in Stop mode

This bit is used to obtain the best trade-off between low-power consumption and restart time when exiting from Stop mode.

When it is set, the flash memory enters low-power mode when the system is in Stop mode.

0: flash memory remains in normal mode when the system enters Stop mode (quick restart time).

1: flash memory enters low-power mode when the system enters Stop mode (low-power consumption).

Note: When system enters Stop mode with SVOS5 enabled, flash memory is automatically forced in low-power mode.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **CSSF**: clear Standby and Stop flags (always read as 0)

This bit is cleared to 0 by hardware.

0: no effect

1: STOPF and SBF flags cleared

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:2 **SVOS[1:0]**: system Stop mode voltage scaling selection

These bits control the V_{CORE} voltage level in system Stop mode, to obtain the best trade-off between power consumption and performance.

00: reserved

01: SVOS5 scale 5

10: SVOS4 scale 4

11: SVOS3 scale 3 (default)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **LPMS**: low-power mode selection

This bit defines the Deepsleep mode.

0: keeps Stop mode when entering DeepSleep.

1: allows Standby mode when entering DeepSleep.

10.11.3 PWR status register (PWR_PMSR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SBF**: System standby flag

This bit is set by hardware and cleared only by a POR or by setting the CSSF bit.

0: system has not been in Standby mode.

1: system has been in Standby mode.

Bit 5 **STOPF**: Stop flag

This bit is set by hardware and cleared only by any reset or by setting the CSSF bit.

0: system has not been in Stop mode.

1: system has been in Stop mode.

Bits 4:0 Reserved, must be kept at reset value.

10.11.4 PWR voltage scaling control register (PWR_VOSCR)

Some register fields are protected against non-secure access, depending on RCC_SECCFGR register. These fields can be protected against unprivileged access, depending on PWR_PRIVCFGR register configuration.

Address offset: 0x0010

Reset value: 0x0000 0000

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

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

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

Bits 5:4 **VOS[1:0]**: voltage scaling selection according to performance

These bits control the V_{CORE} voltage level and allow to obtain the best trade-off between power consumption and performance:

- In bypass mode, these bits must also be set according to the external provided core voltage level and related performance.
- When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
- When decreasing performance, the system frequency must first be decreased before changing the voltage scaling.

00: scale 3 (default)

01: scale 2

10: scale 1

11: scale 0

Bits 3:0 Reserved, must be kept at reset value.

10.11.5 PWR voltage scaling status register (PWR_VOSSR)

Address offset: 0x0014

Reset value: 0x0000 2008

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTVOS[1:0]		ACTVOS RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS RDY	Res.	Res.	Res.
r	r	r										r			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ACTVOS[1:0]**: voltage output scaling currently applied to V_{CORE}
This field provides the last VOS value.
00: VOS3 (lowest power)
01: VOS2
10: VOS1
11: VOS0 (highest frequency)

Bit 13 **ACTVOSRDY**: Voltage level ready for currently used VOS
0: V_{CORE} is above or below the current voltage scaling provided by ACTVOS[1:0]
1: V_{CORE} is equal to the current voltage scaling provided by ACTVOS[1:0]

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **VOSRDY**: Ready bit for V_{CORE} voltage scaling output selection.
0: Not ready, voltage level below VOS selected level
1: Ready, voltage level at or above VOS selected level
The VOSRDY flag must be used only when switching from low to high-voltage scale (like switching from VOS3 to VOS0).

Bits 2:0 Reserved, must be kept at reset value.

10.11.6 PWR Backup domain control register (PWR_BDCR)

This register is protected against non-secure access when VBSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when VBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when VBSEC = 0 and NSPRIV = 1.

This register is not reset by wake-up from Standby mode, RESET signal and VDD POR. It is reset only by VSW POR and VSWRST reset. This register must not be accessed when VSWRST bit in RCC_BDCR register resets the VSW domain.

MONEN and BREN bits must not be accessed when VSWRST bit in RCC_BDCR register resets the VSW domain. After reset, MONEN and BREN of this register are write-protected.

Prior to modifying their content, the DBP bit in PWR_DBPCR register must be set to disable the write protection.

Address offset:0x20

Power-on reset value: 0x0000 0000

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

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

Bit 9 **VBRS**: V_{BAT} charging resistor selection
0: Charge V_{BAT} through a 5 kΩ resistor.
1: Charge V_{BAT} through a 1.5 kΩ resistor.

Bit 8 **VBE**: V_{BAT} charging enable
0: V_{BAT} battery charging disabled.
1: V_{BAT} battery charging enabled.
Note: Reset only by POR.

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

Bit 1 **MONEN**: Backup domain voltage and temperature monitoring enable
0: Backup domain voltage and temperature monitoring disabled
1: Backup domain voltage and temperature monitoring enabled

Bit 0 **BREN**: Backup RAM retention in Standby and V_{BAT} modes
When this bit set, the backup regulator (used to maintain the backup RAM content in Standby and V_{BAT} modes) is enabled.
If BREN is cleared, the backup regulator is switched off. The backup RAM can still be used in Run and Stop modes. However its content is lost in Standby and V_{BAT} modes.
If BREN is set, the application must wait till the backup regulator ready flag (BRRDY) is set to indicate that the data written into the SRAM is maintained in Standby and V_{BAT} modes.
0: Backup RAM content lost in Standby and V_{BAT} modes.
1: Backup RAM content preserved in Standby and V_{BAT} modes

10.11.7 PWR Backup domain control register (PWR_DBPCR)

This register is protected against non-secure access when VBSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when VBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when VBSEC = 0 and NSPRIV = 1.

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bit 0 **DBP**: Disable Backup domain write protection

In reset state, all registers and SRAM in Backup domain are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Write access to Backup domain disabled

1: Write access to Backup domain enabled

10.11.8 PWR Backup domain status register (PWR_BDSR)

This register is not reset by wake-up from Standby mode, RESET signal and VDD POR. It is reset only by VSW POR and VSWRST reset.

Address offset: 0x028

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TEMPH**: temperature level monitoring versus high threshold

0: temperature below high threshold level

1: temperature equal or above high threshold level

Bit 22 **TEMPL**: temperature level monitoring versus low threshold

0: temperature above low threshold level

1: temperature equal or below low threshold level

Bit 21 **VBATH**: V_{BAT} level monitoring versus high threshold

0: V_{BAT} level below high threshold level

1: V_{BAT} level equal or above high threshold level

Bit 20 **VBATL**: V_{BAT} level monitoring versus low threshold
0: V_{BAT} level above low threshold level
1: V_{BAT} level equal or below low threshold level

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **BRRDY**: backup regulator ready
This bit is set by hardware to indicate that the backup regulator is ready.
0: backup regulator not ready
1: backup regulator ready

Bits 15:0 Reserved, must be kept at reset value.

10.11.9 PWR USB Type-C power delivery register (PWR_UCPDR)

This register is protected against non-secure access when UCPD1SEC = 1 in the TZSC_SECCFGR register, and against unprivileged access when UCPD1SEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when UCPD1SEC = 0 and NSPRIV = 1.

Address offset: 0x02C

Reset value: 0x0000 0000

Not affected by exit Standby mode.

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

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

Bit 1 **UCPD_STBY**: USB Type-c and Power delivery Standby mode
When set, this bit is used to memorize the UCPD configuration in Standby mode. This bit must be written to 1 just before entering Standby mode when using UCPD, and it must be written to 0 after exiting the standby mode and before writing any UCPD register.

Bit 0 **UCPD_DBDIS**: USB Type-C and power delivery dead battery disable
After exiting reset, the USB Type-C “dead battery” behavior is enabled, which may have a pull-down effect on CC1 and CC2 pins. It is recommended to disable it in all case, either to stop this pull-down or to hand over control to the UCPD (which should therefore be initialized before doing the disable).
0: Enable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins.
1: Disable USB Type-C dead battery pull-down behavior on UCPDx_CC1 and UCPDx_CC2 pins.

10.11.10 PWR supply configuration control register (PWR_SCCR)

This register is protected against non-secure access when SCMSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when SCMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when SCMSEC = 0 and NSPRIV = 1.

Address offset: 0x030

Reset value: 0x0000 0X00 (STM32H562/563/573xx devices)

Reset value: 0x0000 0100 (STM32H523/533xx devices)

The reset value of this register changes according to the package. Bits 9 and 8 indicate the power configuration. Their values are exclusive.

Reset by POR only, not reset by wake-up from Standby mode and RESET pad. The BYPASS bit of this register is written once after POR. Written-once mechanism locks the register and any further write access is ignored. The system must be power cycled before writing a new value.

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

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

Bit 9 **SMPSEN**: SMPS enable

The value is set by hardware when the package uses the SMPS regulator.

Note: Check the actual availability of this bit on the datasheet. If not available, consider it as reserved, and keep it at reset value.

Bit 8 **LDOEN**: LDO enable

The value is set by hardware when the package uses the LDO regulator.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BYPASS**: power management unit bypass

0: Power management unit normal operation. Use the internal regulator.

1: Power management unit bypassed. Use the external power (voltage monitoring still active)

10.11.11 PWR voltage monitor control register (PWR_VMCR)

This register is protected against non-secure access when SCMSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when SCMSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when SCMSEC = 0 and NSPRIV = 1.

The PVDE and PLS bits are protected by lock mechanism. The lock control is in the SBS module controlled b.y PVDL bit in SBS_CFGR2 register. By default, the PVDE and PLS are unlocked.

Address offset: 0x034

Reset value: 0x0000 0000

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:9 **ALS[1:0]**: analog voltage detector (AVD) level selection
These bits select the analog voltage detector (AVD) thresholds.
00: AVD level0 (V_{AVD0} ~ 1.7 V)
01: AVD level1 (V_{AVD1} ~ 2.1 V)
10: AVD level2 (V_{AVD2} ~ 2.5 V)
11: AVD level3 (V_{AVD3} ~ 2.8 V)

Bit 8 **AVDEN**: peripheral voltage monitor on V_{DDA} enable
0: peripheral voltage monitor on V_{DDA} disabled
1: peripheral voltage monitor on V_{DDA} enabled

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:1 **PLS[2:0]**: programmable voltage detector (PVD) level selection
These bits select the programmable voltage detector (PVD) thresholds.
000: PVD level0 (VPVD0 ~ 1.95 V)
001: PVD level1 (VPVD1 ~ 2.10 V)
010: PVD level2 (VPVD2 ~ 2.25 V)
011: PVD level3 (VPVD3 ~ 2.40 V)
100: PVD level4 (VPVD4 ~ 2.55 V)
101: PVD level5 (VPVD5 ~ 2.70 V)
110: PVD level6 (VPVD6 ~ 2.85 V)
111: PVD_IN pin

Bit 0 **PVDE**: PVD enable
0: PVD disabled
1: PVD enabled

10.11.12 PWR USB supply control register (PWR_USBSCR)

This register is protected against non-secure access when VUSBSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when VUSBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when VUSBSEC = 0 and NSPRIV = 1.

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	USB33 SV	USB33 DEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

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

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

Bit 25 **USB33SV:** independent USB supply valid

This bit is used to validate the V_{DDUSB} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the USBFS peripheral. If V_{DDUSB} is not always present in the application, the V_{DDUSB} voltage monitor can be used to determine whether this supply is ready or not.

0: V_{DDUSB} is not present. Logical and electrical isolation is applied to ignore this supply.

1: V_{DDUSB} is valid.

Bit 24 **USB33DEN**: V_{DDUSB} voltage level detector enable

0: V_{DDUSB} voltage level detector disabled

1: V_{DDUSB} voltage level detector enabled

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

10.11.13 PWR voltage monitor status register (PWR_VMSR)

Address offset: 0x03C

Reset value: 0x00X0 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB33 RDY	Res.	PVDO	Res.	VDDIO2 RDY	AVDO	Res.	Res.	Res.
							r		r		r	r			

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USB33RDY**: V_{DDUSB} ready

0: V_{DDUSB} is below the threshold of the V_{DDUSB} voltage monitor.

1: V_{DDUSB} is equal or above the threshold of the V_{DDUSB} voltage monitor.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **PVDO**: programmable voltage detect output

This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.

0: V_{DD} is equal or higher than the PVD threshold selected through the PLS[2:0] bits.

1: V_{DD} is lower than the PVD threshold selected through the PLS[2:0] bits.

Note: Since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **VDDIO2RDY**: voltage detector output on V_{DDIO2}

This bit is set and cleared by hardware.

0: V_{DDIO2} is below 1.2 V.

1: V_{DDIO2} is above or equal to 1.2 V.

Bit 19 **AVDO**: analog voltage detector output on V_{DDA}

This bit is set and cleared by hardware. It is valid only if AVD on VDDA is enabled by the AVDEN bit.

0: V_{DDA} is equal or higher than the AVD threshold selected with the ALS[2:0] bits.

1: V_{DDA} is lower than the AVD threshold selected with the ALS[2:0] bits.

Note: Since the AVD is disabled in Standby mode, this bit is equal to 0 after standby or reset until the AVDEN bit is set.

Bits 18:0 Reserved, must be kept at reset value.

10.11.14 PWR wake-up status clear register (PWR_WUSCR)

Each register bit CWUFx (x = 1 to 8) is protected against non-secure access when WUPxSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC 0 and NSPRIV = 1.

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF8	CWUF7	CWUF6	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
								w	w	w	w	w	w	w	w

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

Bits 7:0 **CWUFx**: clear wake-up pin flag for WUFx (x = 8 to 1)

These bits are always read as 0.

0: no effect

1: writing 1 clears the WUFx wake-up pin flag (bit is cleared to 0 by hardware).

10.11.15 PWR wake-up status register (PWR_WUSR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
								r	r	r	r	r	r	r	r

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

Bits 7:0 **WUFx**: wake-up pin WUFx flag (x = 8 to 1)

This bit is set by hardware and cleared only by a RESET pin or by setting the CWUFx bit in PWR_WUSCR register.

0: no wake-up event occurred.

1: wake-up event received from WUFx pin.

10.11.16 PWR wake-up configuration register (PWR_WUCR)

Each WUPPUPDx (x = 1 to 8), WUPPx (x = 1 to 8), and WUPENx (x = 1 to 8) bit is protected against non-secure access when WUPxSEC = 1 (x = 1 to 8) in the PWR_SECCFGR register. Each WUPENx bit is protected against unprivileged access when WUPxSEC = 1 in PWR_SECCFGR and SPRIV = 1 in PWR_PRIVCFGR, or when WUPxSEC = 0 and NSPRIV = 1.

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUPPUPD8[1:0]		WUPPUPD7[1:0]		WUPPUPD6[1:0]		WUPPUPD5[1:0]		WUPPUPD4[1:0]		WUPPUPD3[1:0]		WUPPUPD2[1:0]		WUPPUPD1[1:0]	
rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUP P8	WUP P7	WUP P6	WUP P5	WUP P4	WUP P3	WUP P2	WUP P1	WUP EN8	WUP EN7	WUP EN6	WUP EN5	WUP EN4	WUP EN3	WUPE N2	WUP EN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WUPPUPDx[1:0]**: wake-up pin pull configuration for WKUPx (x = 8 to 1)

These bits define the I/O pad pull configuration used when WUPENx = 1. The associated GPIO port pull configuration must be set to the same value or to 00. The wake-up pin pull configuration is kept in Standby mode.

00: no pull-up

01: pull-up

10: pull-down

11: reserved

Bits 15:8 **WUPPx**: wake-up pin polarity bit for WUPx (x = 8 to 1)

These bits define the polarity used for event detection on WUPx external wake-up pin.

0: detection on high level (rising edge)

1: detection on low level (falling edge)

Bits 7:0 **WUPENx**: enable wake-up pin WUPx (x = 8 to 1)

These bits are set and cleared by software.

0: an event on WUPx pin does not wake-up the system from Standby mode.

1: a rising or falling edge on WUPx pin wakes up the system from Standby mode.

Note: An additional wake-up event is detected if WUPx pin is enabled (by setting the WUPENx bit) when WUPx pin level is already high when WUPPx selects rising edge, or low when WUPPx selects falling edge.

10.11.17 PWR I/O retention register (PWR_IORETR)

This register is protected against non-secure access when RETSEC = 1 in the PWR_SECCFGR register, and against unprivileged access when RETBSEC = 1 and SPRIV = 1 in the PWR_PRIVCFGR register, or when RETSEC = 0 and NSPRIV = 1.

Address offset: 0x050

Reset value: 0x0000 0000

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **JTAGIORETEN**: IO retention enable for JTAG IOs

0: IO Retention mode is disabled.

1: IO Retention mode is enabling for PA13, PA14, PA15, and PB4.

when entering into standby mode, the output is sampled, and apply to the output IO during the standby power mode

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **IORETEN**: IO retention enable

0: IO Retention mode is disabled.

1: IO Retention mode is enabling for all IO except the IO support the standby functionality and PA13, PA14, PA15, and PB4.

When entering into standby mode, the output is sampled, and apply to the output IO during the standby power mode.

Note: The IO state is not retained if the DBG_STANDBY bit is set in DBGMCU_CR register.

10.11.18 PWR security configuration register (PWR_SECCFGR)

This register can be written only when the access is secure. It can be read by secure or non-secure access. The register is write-protected against unprivileged write access when SPRIV = 1 in the PWR_PRIVCFGR.

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VUSBSEC	VBSEC	SCMSEC	LPMSEC	RETSEC	Res.	Res.	Res.	WUP8SEC	WUP7SEC	WUP6SEC	WUP5SEC	WUP4SEC	WUP3SEC	WUP2SEC	WUP1SEC
rw	rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **VUSBSEC**: voltage USB secure protection

0: PWR_USBSCR can be read and written with secure or non-secure access.

1: PWR_USBSCR can be read and written only with secure access..

Bit 14 **VBSEC**: Backup domain secure protection
0: PWR_BDCR, PWR_DBPCR can be read and written with secure or non-secure access.
1: PWR_BDCR, PWR_DBPCR can be read and written only with secure access.

Bit 13 **SCMSEC**: supply configuration and monitoring secure protection.
0: PWR_SCCR and PWR_VMCR can be read and written with secure or non-secure access.
1: PWR_SCCR and PWR_VMCR can be read and written only with secure access.

Bit 12 **LPMSEC**: low-power modes secure protection
0: PWR_PMCR can be read and written with secure or non-secure access.
1: PWR_PMCR can be read and written only with secure access.

Bit 11 **RETSEC**: retention secure protection
0: PWR_IORETR can be read and written with secure or non-secure access.
1: PWR_IORETR can be read and written only with secure access.

Bits 10:8 Reserved, must be kept at reset value.

Bits 7:0 **WUPxSEC**: WUPx secure protection (x = 8 to 1)
0: The bits related to the WKUPx wake-up pin in PWR_WUSCR and PWR_WUCR can be read and written with secure or non-secure access.
1: The bits related to the WKUPx wake-up pin in PWR_WUSCR and PWR_WUCR can be read and written only with secure access.

10.11.19 PWR privilege configuration register (PWR_PRIVCFGR)

This register can be written only when the access is privileged. It can be read by privileged or unprivileged access.

Address offset: 0x104

Reset value: 0x0000 0000

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

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

Bit 1 **NSPRIV**: PWR non-secure functions privilege configuration
Set and reset by software. This bit can be written only by privileged access, secure or non-secure.
0: Read and write to PWR non-secure functions can be done by privileged or unprivileged access.
1: Read and write to PWR non-secure functions can be done by privileged access only.

Bit 0 **SPRIV**: PWR secure functions privilege configuration
Set and reset by software. This bit can be written only by a secure privileged access.
0: Read and write to PWR secure functions can be done by privileged or unprivileged access.
1: Read and write to PWR secure functions can be done by privileged access only.

10.11.20 PWR register map

Table 110. PWR register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**PWR_PMCR(1)**	Res.	Res.	Res.	Res.	Res.	SRAM1SO	SRAM2_48SO	SRAM2_16SO	SRAM3SO	Res.	Res.	Res.	Res.	Res.	Res.	ETHERNETSO	Res.	Res.	AVD_READY	BOOSTE	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	SVOS[1:0]		Res.	LPMS
	Reset value						0	0	0	0							0			0	0			0		0				1	1		0
0x00	**PWR_PMCR(2)**	Res.	Res.	Res.	Res.	SRAM1SO	SRAM2_48SO	SRAM2_16HSO	SRAM2_16LO	SRAM3SO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVD_READY	BOOSTE	Res.	Res.	FLPS	Res.	CSSF	Res.	Res.	Res.	SVOS[1:0]		Res.	LPMS
	Reset value						0	0	0	0										0	0			0		0				1	1		0
0x04	**PWR_PMSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0					
0x008 to 0x00F	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x10	**PWR_VOSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOS [1:0]		Res.	Res.	Res.	Res.
	Reset value																											0	0				
0x14	**PWR_VOSSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACTVOS [1:0]		ACTVOSRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VOSRDY	Res.	Res.	Res.
	Reset value																	0	0	1										1			
0x018 to 0x01F	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x20	**PWR_BDCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBRS	VBE	Res.	Res.	Res.	Res.	Res.	Res.	MONEN	BREN
	Reset value																							0	0							0	0
0x24	**PWR_DBPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
	Reset value																																0
0x28	**PWR_BDSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	VBATH	VBATL	Res.	Res.	Res.	BRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0				0																
0x2C	**PWR_UCPDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD_STBY	UCPD_DBDIS
	Reset value																															0	0
0x30	**PWR_SCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPSEN(3)	LDOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BYPASS
	Reset value																							X	X								0

Table 110. PWR register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x34	**PWR_VMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALS [1:0]		AVDEN	Res.	Res.	Res.	Res.	PLS [2:0]			PVDE
	Reset value																						0	0	0							0	0
0x38	**PWR_USBSCR**	Res.	Res.	Res.	Res.	Res.	Res.	USB33SV	USB33DEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value							0	0																								
0x3C	**PWR_VMSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB33RDY	Res.	PVDD	Res.	VDDIO2RDY	AVDD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0		X	0																			
0x40	**PWR_WUSCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CWUF8	CWUF7	CWUF6	CWUF5	CWUF4	CWUF3	CWUF2	CWUF1
	Reset value																									0	0	0	0	0	0	0	0
0x44	**PWR_WUSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUF8	WUF7	WUF6	WUF5	WUF4	WUF3	WUF2	WUF1
	Reset value																									0	0	0	0	0	0	0	0
0x48	**PWR_WUCR**	WUPPUPD8 [1:0]		WUPPUPD7 [1:0]		WUPPUPD6 [1:0]		WUPPUPD5 [1:0]		WUPPUPD4 [1:0]		WUPPUPD3 [1:0]		WUPPUPD2 [1:0]		WUPPUPD1 [1:0]		WUPP8	WUPP7	WUPP6	WUPP5	WUPP4	WUPP3	WUPP2	WUPP1	WUPEN8	WUPEN7	WUPEN6	WUPEN5	WUPEN4	WUPEN3	WUPEN2	WUPEN1
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x50	**PWR_IORETR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JTAGIORETEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IORETEN
	Reset value																0																0
0x054 to 0x0FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x100	**PWR_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VUSBSEC	VBSEC	SCMSEC	LPMSEC	RETSEC	Res.	Res.	Res.	WUP7SEC	WUP6SEC	WUP5SEC	WUP4SEC	WUP3SEC	WUP2SEC	Res.	WUP2SEC
	Reset value																	0	0	0	0	0				0	0	0	0	0	0		0
0x104	**PWR_PRIVCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRIV	
	Reset value																														0	0	

1. STM32H562/572/573xx devices only.
2. STM32H523/533xx devices only.
3. This bit is reserved on STM32H523/533xx devices, keep it at reset value.

Refer to *Section 2.3* for the register boundary addresses.

11 Reset and clock control (RCC)

11.1 Introduction

The reset and clock control (RCC) manages the different resets, and generates the clocks for the bus and peripherals.

11.2 RCC pins and internal signals

Table 111 lists the RCC inputs and output signals connected to package pins or balls.

Table 111. RCC input/output signals connected to package pins or balls

Signal name	Signal type	Description
NRST	I/O	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO	O	Output clock for external devices
LSCO	O	Low-speed output clock for external devices
AUDIOCLK	I	External kernel clock input for SAI1, SAI2, I2S1, I2S2, and I2S3

11.3 RCC reset functional description

There are three types of reset:

- a system reset
- a power reset
- a Backup domain reset

11.3.1 Power reset

A power reset is generated when one of the following events occurs:

- a brownout reset (BOR)
- when exiting Standby mode

A brownout reset, including power-on or power-down reset (POR/PDR), sets all registers to their reset values except the ones in the Backup domain.

When exiting Standby mode, all registers in the core domain are set to their reset value. Registers outside the core domain (RTC, WKUP, IWDG, and GPIO pullup/pulldown configuration during Standby and Standby mode exit) are not impacted.

11.3.2 System reset

A system reset sets all registers to their reset values except the reset flags in *RCC reset status register (RCC_RSR)* and the registers in the Backup domain.

A system reset is generated when one of the following events occurs:

- a low level on the NRST pin (external reset)
- a window watchdog event (WWDG reset)
- an independent watchdog event (IWDG reset)
- a software reset (SW reset) (see *Software reset*)
- a low-power mode security reset (see *Low-power mode security reset*)
- a brownout reset

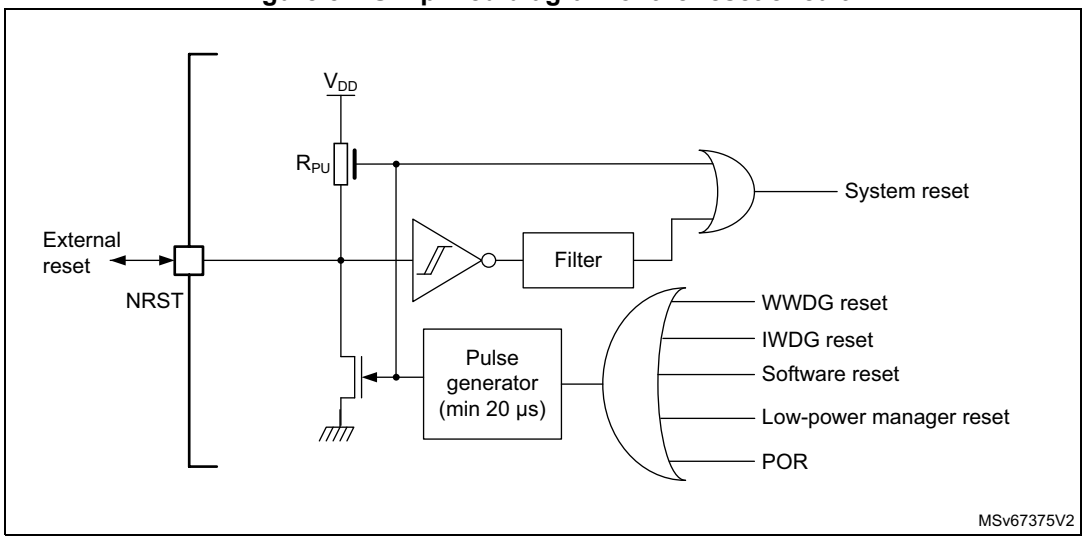
The reset source can be identified by checking the reset flags in *RCC reset status register (RCC_RSR)*.

These sources act on the NRST pin and this pin is always kept low during the delay phase. The reset service routine vector is selected depending on product state, on Boot option bytes or on both.

The system reset signal provided to the device is output on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each internal reset source. In case of an external reset, the reset pulse is generated while the NRST pin is asserted low.

In case on an internal reset, the internal pull-up R_{PU} is deactivated to save the power consumption through the pull-up resistor.

Figure 51. Simplified diagram of the reset circuit



Software reset

The SYSRESETREQ bit in Cortex-M33 application interrupt and reset control register must be set to force a software reset on the device.

Low-power mode security reset

To avoid that critical applications mistakenly enter a low-power mode, the following low-power mode security resets are available. If enabled in option bytes, the resets are generated in any of the following conditions:

- Entering Standby mode: this type of reset is enabled by resetting nRST_STDBY bit in user option bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
- Entering Stop mode: this type of reset is enabled by resetting nRST_STOP bit in user option bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

For further information on the user option bytes, refer to *Section 7.4.1: Option bytes description*.

11.3.3 Backup domain reset

The Backup domain has two specific resets, generated after one of the following events:

- a software reset, triggered by setting the VSWRST bit in the *RCC Backup domain control register (RCC_BDCR)*. Write access to this domain must be enabled before setting VSWRST bit to perform the reset.
- a V_{DD} or V_{BAT} power on, if both supplies have previously been powered off

A Backup domain reset affects the LSE oscillator, the RTC, the backup registers, the backup SRAM, and the RCC_BDCR register.

11.3.4 Reset source identification

The application can identify the reset source by checking the reset flags in the RCC_RSR register. The software can reset the flags by setting the RMVF bit.

Table 112 shows how the status bits of the RCC_RSR register behave according to the situation that generated the reset. For example, when an IWDG timeout occurs, if the CPU is reading the RCC_RSR register during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

Table 112. Reset source identification (RCC_RSR)(1)

	Reset	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF
1	Power-on reset	0	0	0	0	1	1
2	Pin/pad reset	0	0	0	0	0	1
3	Brownout (low or high) reset	0	0	0	0	1	1
4	System reset generated by CPU	0	0	0	1	0	1
5	WWDG reset	0	1	0	0	0	1
6	IWDG reset	0	0	1	0	0	1
7	Illegal stop entry reset	1	0	0	0	0	1

1. Gray cells highlight the register bits that are set.

11.4 RCC clocks functional description

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI: high-speed internal up to 64 MHz RC oscillator clock
- CSI: low power internal RC oscillator clock
- HSE: high-speed external crystal or clock, from 4 to 50 MHz
- PLL1 clock

The HSI is used as system clock source after startup from reset, configured at 32 MHz.

The device has the following additional clock sources:

- LSI: 32 kHz low-speed internal RC that drives the independent watchdog and optionally the RTC used for auto-wakeup from Stop and Standby modes
- LSE: 32.768 kHz low-speed external crystal or clock that optionally drives the real-time clock (rtc_ck)
- HSI48: internal 48 MHz RC that potentially drives the USB FS and the RNG
- PLL2 and PLL3 clocks

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Several prescalers can be used to configure the AHB frequency, the APB1 and APB2 domains. The maximum frequency of the AHB and APB domains is 250 MHz.

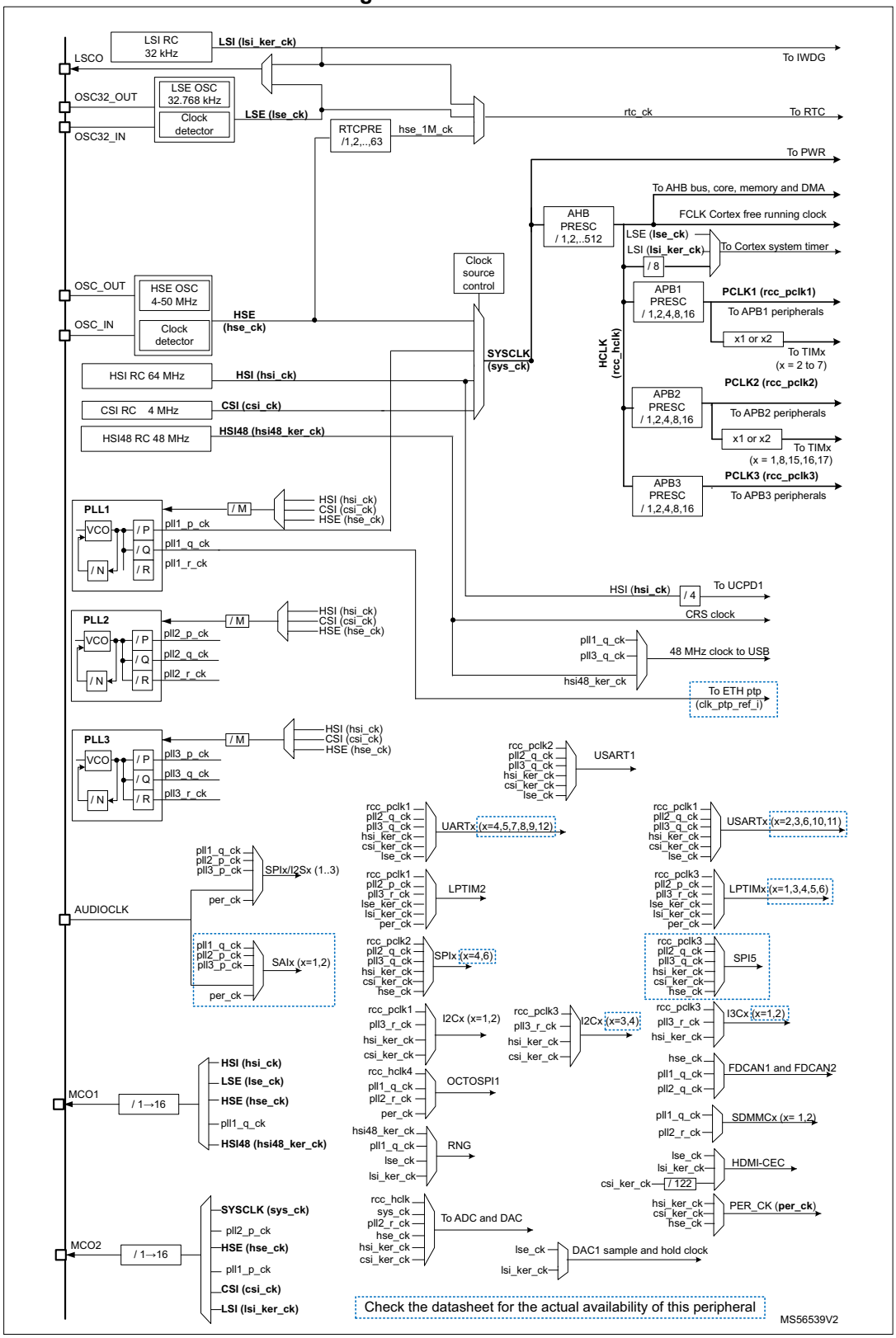
The peripheral clocks are derived from their bus clock (HCLK, PCLK1, PCLK2 or PCLK3), except those that receive an independent kernel clock. This kernel clock can be selected by software between several sources thanks to RCC_CCIPRx registers (x = 1,2,3,4,5).

In addition, the RTC kernel clock is selected by software in RCC_BDCR. The IWDG clock is always the LSI 32 kHz clock.

The RCC feeds the Cortex system timer (SysTick) external clock with the AHB clock (HCLK) divided by eight, or LSE or LSI. The SysTick can work either with this clock or directly with the Cortex clock (HCLK), configurable in the SysTick control and status register.

FCLK acts as Cortex-M33 free-running clock.

Figure 52. Clock tree

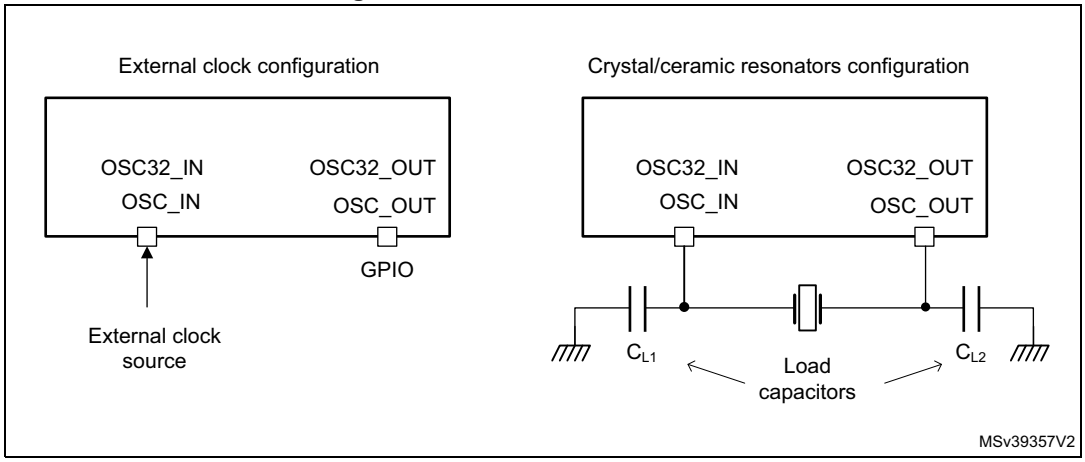


1. For details on the internal/external clock source, refer to the Electrical characteristics in the datasheet.

11.4.1 HSE clock

The HSE block can generate a clock from an external crystal/ceramic resonator, or from an external clock source.

Figure 53. HSE/LSE clock sources



External clock source (HSE bypass)

In this mode, an external clock source must be provided to OSC_IN pin. The external clock can be low swing (analog) or digital. If this clock is directly used by a peripheral, the duty cycle requirement is defined by the peripheral and the application (refer to datasheet for more details).

In case of an analog clock (low swing) the HSEBYP and HSEON bits must be set to 1 in the *RCC clock control register (RCC_CR)*.

In case of a digital clock, the HSEBYP and the HSEEXT bits must be set to 1 followed by setting the HSEON bit to 1 in the *RCC clock control register (RCC_CR)*.

External crystal/ceramic resonator

The oscillator is enabled by setting the HSEBYP bit to 0 and HSEON bit to 1.

The HSE can be used when the product requires a very accurate high-speed clock.

The associated hardware configuration is shown in *Figure 53*: the resonator and the load capacitors must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The HSERDY flag of the *RCC clock control register (RCC_CR)* indicates whether the HSE oscillator is stable or not. At startup, the **hse_ck** clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

The HSE can be switched ON and OFF through the HSEON bit. Note that the HSE cannot be switched OFF if one of the following two conditions is met:

- the HSE is used directly (via software mux) as system clock
- the HSE is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSEON bit to 0.

The HSE is automatically disabled by hardware, when the system enters Stop or Standby mode.

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.

11.4.2 HSI clock

The HSI block provides the default clock to the product.

The HSI is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32 or 64 MHz. This predivider is controlled by the HSIDIV.

The HSI advantages are the following:

- low-cost clock source, as no external crystal is required
- faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The HSI can be switched ON and OFF using the HSION bit. Note that the HSI cannot be switched OFF if one of the two conditions is met:

- the HSI is used directly (via software mux) as system clock
- the HSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the HSION bit to 0. Note that the HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case the hardware does not update the HSIDIV with the new value. However it is possible to change the HSIDIV if the HSI is used directly as system clock.

The HSIRDY flag indicates if the HSI is stable or not. At startup, the HSI output clock is not released until this bit is set by hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to *Section 11.4.10: Clock security system (CSS)*). The HSI can be disabled or not when the system enters Stop mode.

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals, the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available
- the frequency accuracy.

Note: The HSI can remain enabled when the system is in Stop mode.

HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACCHSI (refer to the product datasheet for more information).

After a power-on reset, the factory calibration value is loaded in the HSICAL[11:0] bits. If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[6:0] bits.

Note: HSICAL[11:0] and HSITRIM[6:0] are located in the RCC CSI calibration register (RCC_CSICFGR).

11.4.3 CSI oscillator

The CSI is a low-power RC oscillator that can be used directly as system clock, peripheral clock, or PLL input.

The CSI advantages are the following:

- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)
- very low-power consumption,

The CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

CSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

The CSI can be switched ON and OFF through the CSION bit. The CSIRDY flag indicates whether the CSI is stable or not. At startup, the CSI output clock is not released until this bit is set by hardware.

The CSI cannot be switched OFF if one of the two conditions is met:

- The CSI is used directly (via software mux) as system clock.
- The CSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).

In that case the hardware does not allow programming the CSION bit to 0.

The CSI can be disabled or not when the system enters Stop mode.

In addition, the CSI clock can be driven to the MCO2 output and used as clock source for other application components.

Even if the CSI settling time is faster than the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency precision.

Note: CSION and CSIRDY bits are located in the RCC clock control register (RCC_CR).

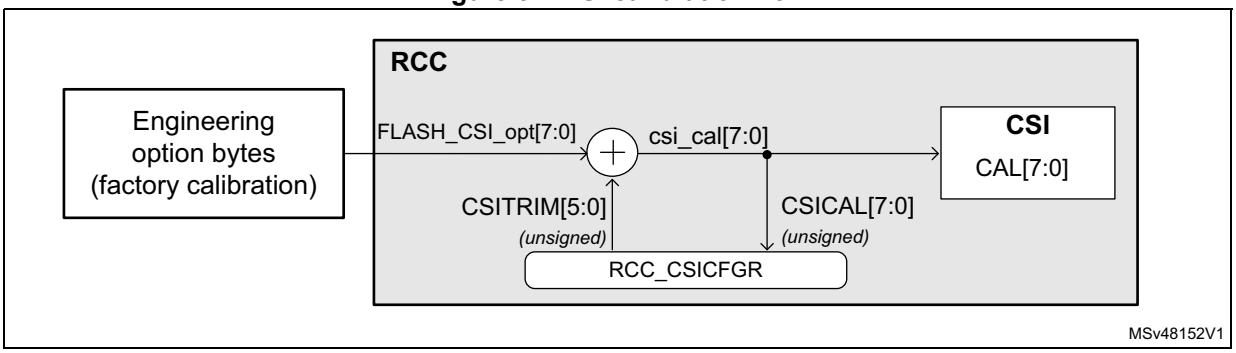
CSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated to achieve ACC_{CSI} accuracy (refer to the product datasheet for more information). After reset, the factory calibration value is loaded in the CSICAL[7:0] bits.

Voltage and/or temperature variations affect the RC oscillator frequency. The user application can trim the CSI frequency using the CSITRIM[5:0] bits.

Note: Bits CSICAL[7:0] and CSITRIM[5:0] are located into the RCC CSI calibration register (RCC_CSICFGR).

Figure 54. CSI calibration flow



11.4.4 HSI48 clock

The HSI48 clock signal is generated from an internal 48 MHz RC oscillator, and can be used directly for USB and for random number generator (RNG).

The internal 48 MHz RC oscillator is mainly dedicated to provide a high-precision clock to the USB peripheral by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE, or an external signal to automatically and quickly adjust the oscillator frequency on-the-fly. It is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, the HSI48 RC oscillator runs on its default frequency, subject to manufacturing process variations.

For more details on how to configure and use the CRS peripheral, refer to *Section 12: Clock recovery system (CRS)*.

The HSI48RDY flag in the RCC_CR register indicates whether the HSI48 RC oscillator is stable or not. At startup, the HSI48 RC oscillator output clock is not released until this bit is set by hardware.

The HSI48 can be switched on and off using the HSI48ON bit in the RCC_CR register.

11.4.5 PLL description

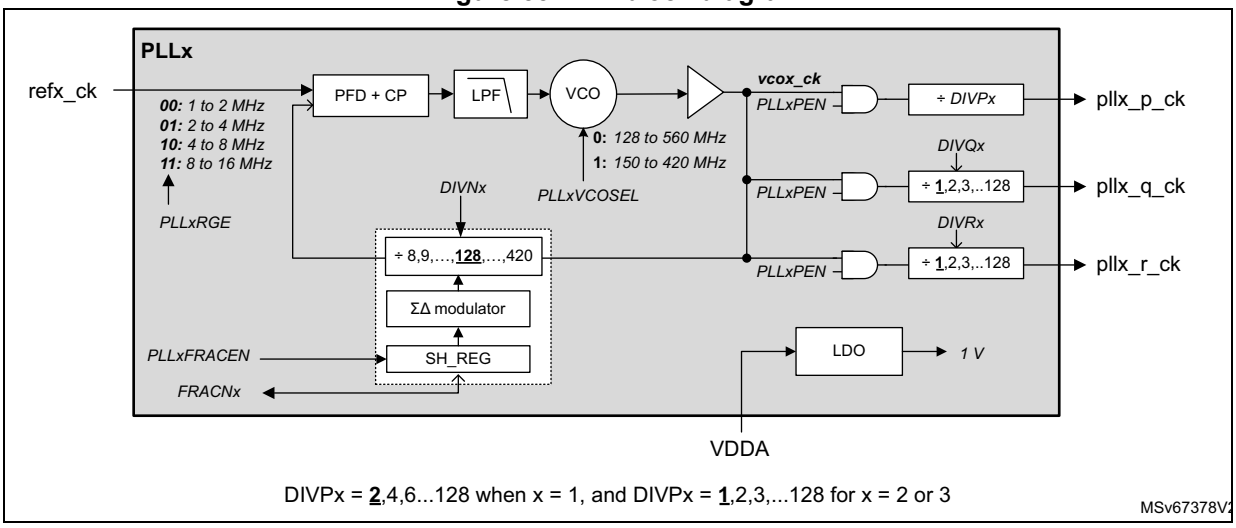
The RCC features three PLLs:

- a main PLL (PLL1), generally used to provide clocks to the CPU and some peripherals
- two dedicated PLLs (PLL2 and PLL3), used to generate the kernel clock for peripherals

The PLLs integrated into the RCC are completely independent. They offer the following features:

- A VCO supporting two modes:
  - a wide-range
  - a low-range used, for instance, in audio applications
- Input frequency range:
  - 2 to 16 MHz for the VCO in wide-range mode
  - 1 to 2 MHz for the VCO in low-range mode
- Capability to work either in integer or fractional mode
- 13-bit sigma-delta modulator, to fine-tune the VCO frequency by 11 to 0.3 ppm steps
- The sigma-delta modulator can be updated on-the-fly without generating frequency overshoots on PLLs outputs
- Each PLL offers three outputs with post-dividers

Figure 55. PLL block diagram



The PLLs are controlled via RCC_PLLxDIVR, RCC_PLLxFRACR, RCC_PLLCFGR, and RCC_CR registers.

The frequency of the reference clock provided to the PLLs (**refx_ck**) must range from 1 to 16 MHz. The PLLxM dividers of the RCC PLLx clock source selection register (RCC_PLLxCFGR) must be properly programmed to match this condition. In addition, the PLLxRGE[1:0] field of the RCC PLLx clock source selection register (RCC_PLLxCFGR) must be set according to the reference input frequency to optimize performance.

The user application can then configure the VCO. The smaller range (150 to 420 MHz) must be chosen when the reference clock frequency is lower than 2 MHz.

To reduce the power consumption, it is recommended to configure the VCO output to the smaller range.

DIVNx loop divider must be programmed to achieve the expected frequency at VCO output. In addition, the VCO output range must be respected.

The PLLs operate in integer mode when the value of SH_REG bit of the FRACNx shadow register is set to 0. The SH_REG bit is updated with the FRACNx value when PLLxFRACEN

bit goes from 0 to 1. The sigma-delta modulator is designed to minimize the jitter impact, while allowing very small frequency steps.

The PLLs can be enabled by setting PLLxON to 1. The PLLxRDY bits indicate that the PLL is ready (locked).

*Note: Before enabling the PLLs, make sure that the reference frequency (**refx_ck**) provided to it is stable, so the hardware does not allow changing PLLxM when the PLLx is ON, and it is also not possible to change PLLSRC when one of the PLLs is ON.*

The hardware prevents writing PLL1ON to 0 if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to HSE clock, HSI clock, and CSI oscillator).

The following PLL parameters cannot be changed once the PLL is enabled: DIVNx, PLLxRGE, PLLxVCOSEL, PLLxP (DIVP), PLLxQ (DIVQ), and PLLxR (DIVR).

For optimal behavior of the PLL when one of the post-divider (DIVP, DIVQ or DIVR) is not used, the application must set the enable bit (DIVyEN) as well as the corresponding post-divider bits (DIVP, DIVQ or DIVR) to 0.

If the above rules are not respected, the PLL output frequency is not guaranteed.

Output frequency computation

When the PLL is configured in integer mode (SH_REG = 0), the VCO frequency (F_{VCO}) is given by the following expression:

$$F_{VCO} = F_{REF_CK} \times DIVN$$

$$F_{PLL_y_CK} = (F_{VCO} / (DIVy + 1)) \text{ with y = P, Q, or R}$$

When the PLL is configured in fractional mode (SH_REG different from 0), the DIVN divider must be initialized before enabling the PLLs. However, it is possible to change the value of FRACNx on-the-fly without disturbing the PLL output.

This feature can be used either to generate a specific frequency from any crystal value with a good accuracy, or to fine-tune the frequency on-the-fly.

For each PLL, the VCO frequency is given by the following formula:

$$F_{VCO} = F_{ref_ck} \times \left(DIVN + \frac{FRACN}{2^{(13)}} \right)$$

Note: For PLL1, DIVP can only take odd values.

The PLLs are disabled by hardware when:

- the system enters Stop or Standby mode
- an HSE failure occurs when HSE or PLL (clocked by HSE) are used as system clock

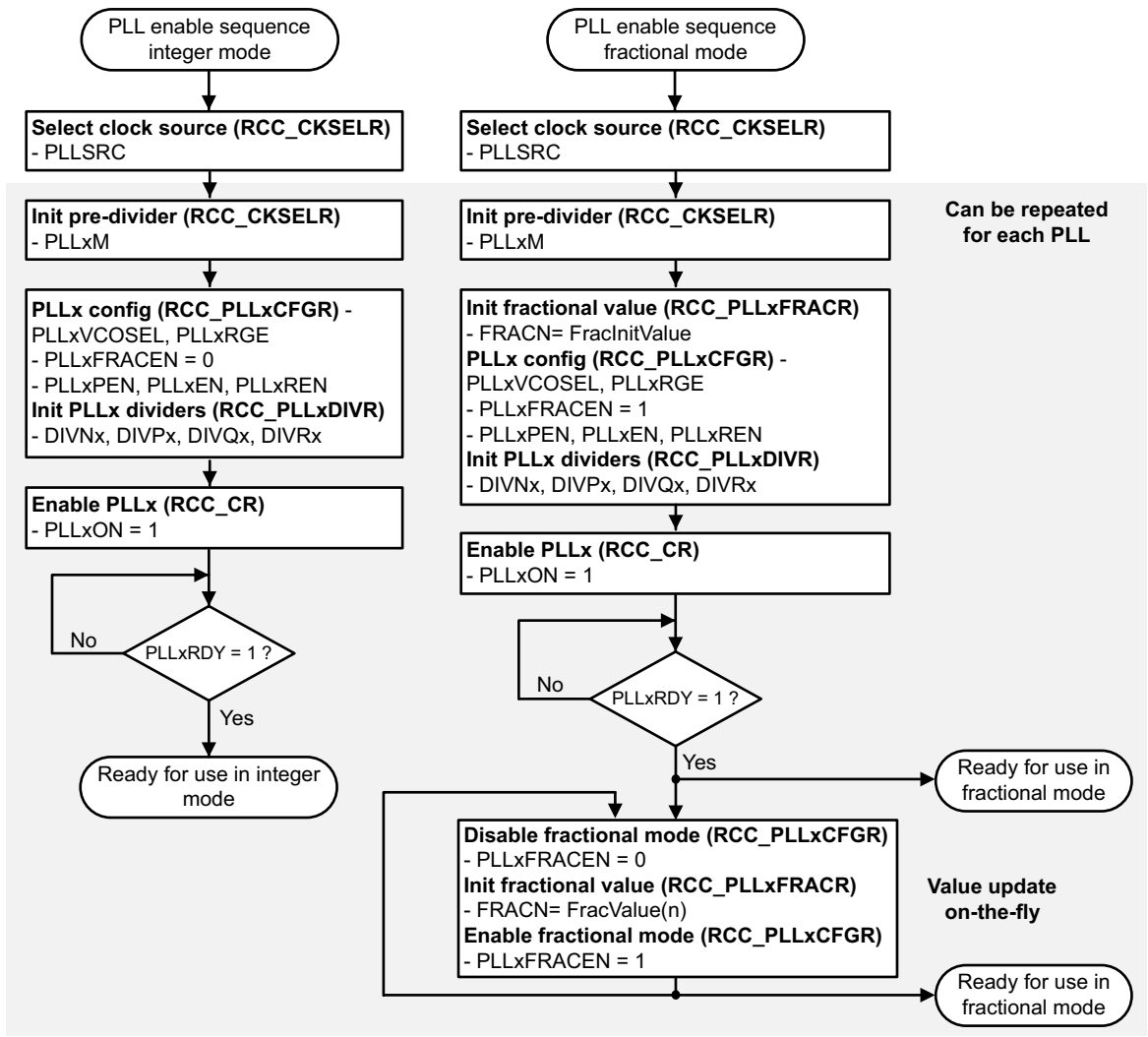
PLL initialization phase

Figure 56 shows the recommended PLL initialization sequence in integer and fractional mode. The PLLx are supposed to be disabled at the start of the initialization sequence:

1. Initialize the PLLs registers according to the required frequency.
   – Set PLLxFRACEN of RCC PLLx clock source selection register (RCC_PLLxCFGR) to 0 for integer mode.
   – For fractional mode, set FRACN to the required initial value (FracInitValue) and then set PLLxFRACEN to 1.
2. Once the PLLxON bit is set to 1, the user application must wait until PLLxRDY bit is set to 1. If the PLLx is in fractional mode, the PLLxFRACEN bit must not be set back to 0 as long as PLLxRDY = 0.
3. Once the PLLxRDY bit is set to 1, the PLLx is ready to be used.
4. If the application intends to tune the PLLx frequency on-the-fly (possible only in fractional mode), then:
   a) PLLxFRACEN must be set to 0. When PLLxFRACEN = 0, the sigma-delta modulator is still operating with the value latched into SH_REG. The application must wait for three clock periods of refx_ck (PLLxFRACEN bit propagation delay).
   b) A new value must be uploaded into PLLxFRACR (FracValue(n)).
   c) PLLxFRACEN must be set to 1, to latch the content of PLLxFRACR into its shadow register.The new value is considered after three clock periods of refx_ck (PLLxFRACEN bit propagation delay).

Note: When the PLLxRDY goes to 1 the difference between the PLLx output frequency and the target value is lower than ±2%.

Figure 56. PLLs initialization flow



11.4.6 LSE clock

The LSE can generate a clock from an external crystal/ceramic resonator, or from an external user clock.

External clock source (LSE bypass)

In this mode, an external clock source must be provided to OSC32_IN pin (see *Figure 53*). The input clock can have a frequency up to 1 MHz, and be low swing (analog) or digital. A duty cycle close to 50% is recommended.

In case of an analog clock (low swing), the LSEBYP and LSEON bits must be set to 1 (*RCC Backup domain control register (RCC_BDCR)*).

In case of a digital clock, the LSEBYP and the LSEEXT bits must be set to 1 followed by setting the LSEON bit to 1 (*RCC Backup domain control register (RCC_BDCR)*). If the RTC

is used, the LSE bypass must not be configured in digital mode, but in low swing analog mode (default value after reset).

External crystal/ceramic resonator (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power, highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions.

The LSERDY flag of the *RCC Backup domain control register (RCC_BDCR)*) indicates whether the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

The LSE oscillator is switched ON and OFF using the LSEON bit. The LSE remains enabled when the system enters Stop or Standby mode.

In addition, the LSE clock can be driven to the MCO1 output and used as clock source for other application components.

The LSE also offers a programmable driving capability (LSEDRV[1:0]), which can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to ensure a stable oscillation.

The driving capability must be set before enabling the LSE oscillator.

11.4.7 LSI clock

The LSI acts as a low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU).

The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced ON and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode.

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the *RCC clock source interrupt enable register (RCC_CIER)*.

In addition, the LSI clock can be driven to the MCO2 output, and used as a clock source for other application components.

Note: Bits LSION and LSIRDY are located in the RCC Backup domain control register (RCC_BDCR).

11.4.8 System clock (SYSCLK) selection

Four different clock sources can be used to drive the system clock (SYSCLK):

- HSI oscillator
- CSI oscillator
- HSE oscillator
- PLL

The system clock maximum frequency is 250 MHz. After a system reset (or after leaving Standby mode), the HSI oscillator, at 32 MHz, is selected as system clock. When a clock source is used directly or through the PLL as a system clock, it is not possible to stop it.

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source not yet ready is selected, the switch occurs when the clock source becomes ready. Status bits in the *RCC clock control register (RCC_CR)* indicate which clocks are ready ,and which clock is currently used as a system clock.

11.4.9 Handling clock generators in stop and standby modes

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped, as well as the following clock sources:

- CSI, HSI (depending on HSIKERON and CSIKERON bits)
- HSE
- PLL1, PLL2, and PLL3
- HSI48

The content of the RCC registers is not altered except for PLL1ON, PLL2ON, PLL3ON, HSEON, and HSI48ON, set to 0.

Exiting Stop mode

When the system exits this mode via a wake-up event, the application can select which oscillator (HSI and/or CSI) is used to restart. The STOPWUCK bit selects the oscillator used as system clock. The STOPKERWUCK bit selects the oscillator used as kernel clock for peripherals. The STOPKERWUCK bit is useful if after a system Stop, a peripheral needs a kernel clock generated by an oscillator different from the one used for the system clock.

These bits belong to the *RCC clock configuration register1 (RCC_CFGR1)*

Table 113. STOPWUCK and STOPKERWUCK description

STOPWUCK	STOPKERWUCK	Activated oscillator(s) when the system exits Stop mode	Distributed clocks when the system exits Stop mode	
			System clock	Kernel clock
0	0	HSI	HSI	HSI
	1	HSI and CSI		HSI and/or CSI
1	0		CSI	
	1	CSI		CSI

During Stop mode

There are two specific cases where the HSI or CSI can be enabled during this mode.

- When a dedicated peripheral requests the kernel clock the peripheral receives the HSI or CSI according to the kernel clock source selected for this peripheral (via CKPERSEL[1:0]).
- When the HSIKERON or CSIKERON bits of the *RCC clock control register (RCC_CR)* are set, the HSI and CSI are kept running, but the outputs are gated. The clock is then

available immediately when the system exits Stop mode, or when a peripheral requests the kernel clock (see *Table 111* for details).

Table 114. HSIKERON and CSIKERON behavior

HSIKERON (CSIKERON)	HSI (CSI) state during Stop mode	HSI (CSI) state setting time
0	OFF	$t_{su(HSI)}$ $t_{su(CSI)}$ (1)
1	Running and gated	Immediate

1. $t_{su(HSI)}$ and $t_{su(CSI)}$ are the startup times of, respectively, the HSI and CSI oscillators (refer to the product datasheet for the values of these parameters).

When the microcontroller exists system standby mode, the HSI is selected as system and kernel clock. The RCC registers are reset to their initial values except for the RCC_RSR and RCC_BDCR registers.

Note: The HSI and CSI outputs provide two clock paths:

- one path for the system clock (hsi_ck or csi_ck)
- one path for the peripheral kernel clock (hsi_ker_ck or csi_ker_ck).

When a peripheral requests the kernel clock in system stop mode, only the path providing the hsi_ker_ck or csi_ker_ck is activated.

11.4.10 Clock security system (CSS)

Clock security system on HSE

The clock security system can be enabled by software via the HSECSSON bit, which can be enabled even when the HSEON is set to 0.

The CSS on HSE is enabled by the hardware when the HSE is enabled and ready, and HSECSSON set to 1.

The CSS on HSE is disabled when the HSE is disabled. As a result, this function does not work when the system is in Stop mode.

It is not possible to clear directly the HSECSSON bit by software.

The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode.

If a failure is detected on the HSE clock, the system automatically switches to the HSI or CSI, depending on STOPWUCK bit configuration in *RCC clock configuration register1 (RCC_CFGR1)*, to provide a safe clock. The HSE is then automatically disabled, a clock failure event is sent to the break inputs of the advanced-control timers (TIM1, TIM8), and an NMI is automatically generated to inform the application about the failure, allowing the MCU to perform rescue operations. If the HSE output was used as clock source for PLLs when the failure occurred, the PLLs are also disabled.

If an HSE clock failure occurs when the CSS is enabled, the CSS generates an interrupt that causes the automatic generation of an NMI. The HSECSSF flag in *RCC clock source interrupt flag register (RCC_CIFR)* is set to 1 to allow the application to identify the failure source. The NMI routine is executed indefinitely until the HSECSSF bit is cleared. As a consequence, the application must clear the HSECSSF flag in the NMI ISR by setting the HSECSSC bit in the *RCC clock source interrupt clear register (RCC_CICR)*.

Clock security system on LSE

A clock security system on LSE can be activated by software writing the LSECSSON bit in the *RCC Backup domain control register (RCC_BDCR)*. This bit can be disabled only by a hardware or RTC software reset, or after a failure detection on LSE. LSECSSON must be written after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware), and after the RTC clock has been selected by RTCSEL.

The CSS operates in all modes, including VBAT. It works also under system reset (excluding power-on reset). The system detects when the LSE disappears, or if it is over frequency.

If a failure is detected on the external 32 kHz oscillator, the LSE clock is no longer supplied to the RTC, but no hardware action is made to the registers.

The CSS on LSE detection event is connected to the internal tamper 3 of the TAMP peripheral. The internal tamper 3 must be enabled (ITAMP3E = 1 in TAMP_CR1 register) and the associated interrupt enabled (ITAMP3IE in TAMP_IER) to wake up from the low-power modes. This erases also the TAMP backup registers and backup SRAM unless the ITAMP3NOER = 1 in the TAMP_CR3 (see *Section 47: Tamper and backup registers (TAMP)* for more details).

In case of CSS on LSE detection event (LSECSSD = 1 in the RCC_BDCR), the software must disable the LSECSSON bit, stop the defective 32 kHz oscillator (disabling LSEON), and change the RTC clock source (no clock or LSI or HSE, with RTCSEL), or take actions to secure the application.

Refer to datasheet for CSS on LSE electrical characteristics.

11.4.11 Clock output generation (MCO1/MCO2)

Two microcontroller clock output pins (MCO1 and MCO2) are available. A clock source can be selected for each output.The selected clock can be divided thanks to configurable prescaler (refer to *Figure 52* for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1[2:0], MCO2PRE[3:0], and MCO2[2:0], located in the *RCC clock configuration register1 (RCC_CFGR1)*.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The clock provided to the MCOs outputs must not exceed the maximum pin speed (refer to the product datasheet for information on the supported speed).

Another output (LSCO) allows one of the low-speed clocks (LSI, LSE) to be output onto the external LSCO pin. This output is available in Stop mode, not available in Standby and VBAT modes. The selection is controlled by the LSCOSEL bit, and enabled by the LSCOEN bit in the *RCC Backup domain control register (RCC_BDCR)*.

The MCO clock output requires the corresponding alternate function selected on the MCO pin. The LSCO pin must be left in default POR state.

11.4.12 Kernel clock selection

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- a clock domain synchronous with the register and bus interface (**ckg_bus_perx** clock)
- a clock domain generally synchronous with the peripheral (kernel clock)

The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral. The user application can thus change the bus frequency without reprogramming the peripherals. As an example, an ongoing transfer with UART is not disturbed if its APB clock is changed on-the-fly.

Table 115 shows the kernel clock that the RCC can deliver to the peripherals. Each row represents a multiplexer and the peripherals connected to its output.

Table 115. Kernel clock distribution overview

Peripherals(1)	Clock multiplexer control bits	pll1_q_ck	pll2_p_ck	pll2_q_ck	pll2_r_ck	pll3_p_ck	pll3_q_ck	pll3_r_ck	sys_ck	bus clocks(2)	hse_ck	hsi_ker_ck	csi_ker_ck	hsi48_ck	lse_ck	lsi_ck	per_ck(3)	AUDIOCLK	Disabled
OCTOSPI	OSPISEL	1	-	-	2	-	-	-	-	0	-	-	-	-	-	-	3	-	-
SDMMC1(4)	SDMMC1SEL	0	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SDMMC2(4)	SDMMC2SEL	0	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-
FDCAN	FDCANSEL	1	-	2	-	-	-	-	-	-	0	-	-	-	-	-	-	-	-
CEC	CECSEL	-	-	-	-	-	-	-	-	-	-	-	2(5)	-	0	1	-	-	-
I2C1	I2C1SEL	-	-	-	-	-	-	1	-	0	-	2	3	-	-	-	-	-	-
I2C2	I2C2SEL	-	-	-	-	-	-	1	-	0	-	2	3	-	-	-	-	-	-
I2C3	I2C3SEL	-	-	-	-	-	-	1	-	0	-	2	3	-	-	-	-	-	-
I2C4	I2C4SEL	-	-	-	-	-	-	1	-	0	-	2	3	-	-	-	-	-	-
I3C1	I3C1SEL	-	-	-	-	-	-	1	-	0	-	2	-	-	-	-	-	-	-
I3C2	I3C2SEL	-	-	-	-	-	-	1	-	0	-	2	-	-	-	-	-	-	-
LPTIM1	LPTIM1SEL	-	1	-	-	-	-	2	-	0	-	-	-	-	3	4	5	-	-
LPTIM2	LPTIM2SEL	-	1	-	-	-	-	2	-	0	-	-	-	-	3	4	5	-	-
LPTIM3	LPTIM3SEL	-	1	-	-	-	-	2	-	0	-	-	-	-	3	4	5	-	-
LPTIM4	LPTIM4SEL	-	1	-	-	-	-	2	-	0	-	-	-	-	3	4	5	-	-
LPTIM5	LPTIM5SEL	-	1	-	-	-	-	2	-	0	-	-	-	-	3	4	5	-	-
LPTIM6	LPTIM6SEL	-	1	-	-	-	-	2	-	0	-	-	-	-	3	4	5	-	-
TIM[8:1],	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-	-
TIM[17:12]	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-	-
TIM16/17	TIMICSEL	-	-	-	-	-	-	-	-	-	-	1	1	-	-	-	-	-	0
RNG	RNGSEL	1	-	-	-	-	-	-	-	-	-	-	-	0	2	3	-	-	-
SAI1	SAI1SEL	0	1	-	-	2	-	-	-	-	-	-	-	-	-	-	4	3	-
SAI2	SAI2SEL	0	1	-	-	2	-	-	-	-	-	-	-	-	-	-	4	3	-
SPI(I2S)1	SPI1SEL	0	1	-	-	2	-	-	-	-	-	-	-	-	-	-	4	3	-
SPI(I2S)2	SPI2SEL	0	1	-	-	2	-	-	-	-	-	-	-	-	-	-	4	3	-
SPI(I2S)3	SPI3SEL	0	1	-	-	2	-	-	-	-	-	-	-	-	-	-	4	3	-

Table 115. Kernel clock distribution overview (continued)

Peripherals[1]	Clock multiplexer control bits	pll1_q_ck	pll2_p_ck	pll2_q_ck	pll2_r_ck	pll3_p_ck	pll3_q_ck	pll3_r_ck	sys_ck	bus clocks[2]	hse_ck	hsi_ker_ck	csi_ker_ck	hsi48_ck	lse_ck	lsi_ck	per_ck[3]	AUDIOCLK	Disabled
SPI4	SPI4SEL	-	-	1	-	-	2	-	-	0	5	3	4	-	-	-	-	-	-
SPI5	SPI5SEL	-	-	1	-	-	2	-	-	0	5	3	4	-	-	-	-	-	-
SPI6	SPI6SEL	-	-	1	-	-	2	-	-	0	5	3	4	-	-	-	-	-	-
USARTx	USARTxSEL	-	-	1	-	-	2	-	-	0	-	3	4	-	5	-	-	-	-
UARTx	UARTxSEL	-	-	1	-	-	2	-	-	0	-	3	4	-	5	-	-	-	-
LPUART1	LPUART1SEL	-	-	1	-	-	2	-	-	0	-	3	4	-	5	-	-	-	-
USB	USBSEL	1	-	-	-	-	2	-	-	-	-	-	-	3	-	-	-	-	0
ADCDAC[6]	ADCDACSEL	-	-	-	2	-	-	-	1	0	3	4	5	-	-	-	-	-	-
DAC	DACSEL	-	-	-	-	-	-	-	-	-	-	-	-	-	0	1	-	-	-
UCPD1	-	-	-	-	-	-	-	-	-	-	-	x[7]	-	-	-	-	-	-	-
ETH (ptp)	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RTC/AWU	RTCSEL	-	-	-	-	-	-	-	-	-	3[8]	-	-	-	1	2	-	-	0
All	CKPERSEL	-	-	-	-	-	-	-	-	-	2	0	1	-	-	-	-	-	3

1. For the actual availability of the peripheral refer to the device datasheet.
2. The bus (APB or AHB) clocks are the bus interface clocks to whom the peripherals are connected.
3. The per_ck clock can be hse_ck, hsi_ker_ck, or csi_ker_ck, according to CKPERSEL selection.
4. With a duty cycle close to 50%, meaning that PLL1Q or PLL2R values must be even. For SDMMCx the duty cycle must be 50% when supporting DDR.
5. Clock CSI divided by 122.
6. With a duty cycle close to 50%, meaning that PLL2R values must be even
7. Clock HSI divided by 4.
8. Clock HSE divided by RTCPRE.

To reduce the number of switches, some peripherals share the same kernel clock source, but all have their dedicated enable signal.

Peripherals dedicated to audio applications

The audio peripherals generally need specific accurate frequencies, the kernel clock of the SAIs or SPI(I2S)s can be generated by:

- PLL1 when the amount of active PLLs must be reduced (for SAIs and SPI1 to 3)
- APB2 peripheral clock (for SPI4 and 5)
- APB3 peripheral clock (for SPI6)
- PLL2 or 3 for optimal flexibility in frequency generation
- HSE, HSI, or CSI when the current consumption is critical
- AUDIOCLK when an external clock reference must be used

Peripherals dedicated to control and data transfer

Peripherals such as SPIs, I2Cs, UARTs do not need a specific kernel clock frequency but a clock fast enough to generate the correct baud rate, or the required bit clock on the serial interface. For that purpose the source can be selected among the following ones:

- PLL1 when the amount of active PLLs must be reduced
- PLL2 or PLL3 if better flexibility is required. As an example, this solution allows changing the frequency bus via PLL1 without affecting the speed of some serial interfaces.
- HSI or CSI for low-power use-cases or when the peripheral must quickly wake up from Stop mode (such as UART or I2C)

Note: UARTs also need the LSE clock when high baud rates are not required.

The OCTOSPI and SDMMC1/2 can also use a clock different from the bus interface one for more flexibility.

RTC/AWU clock

The **rtc_ck** clock source can be one of the following:

- the **hse_1M_ck** (**hse_ck** divided by a programmable prescaler)
- the **lse_ck**
- the **lsi_ck** clock

The source clock is selected by programming the RTCSEL[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)* and the RTCPRE[5:0] bits in the *RCC clock configuration register1 (RCC_CFGR1)*.

This selection cannot be modified without resetting the Backup domain.

If the LSE is selected as RTC clock, the RTC works normally even if the backup or the V_{DD} supply disappears.

The LSE clock is in the Backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as RTC clock, the RTC continues working even if the V_{DD} supply is switched OFF, provided the V_{BAT} supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the V_{CORE} supply is powered off.

The **rtc_ck** clock is enabled through RTCEN bit located in the *RCC Backup domain control register (RCC_BDCR)*.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLPEN bits located in RCC_APB3ENR/LPENR registers.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency ($F_{APB} < 7 \times F_{RTCLCK}$), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result than the first one. Otherwise a third read access must be performed.

Watchdog clocks

The RCC provides the clock for the two watchdog blocks available on the circuit. The independent watchdog (IWDG) is connected to the LSI. The window watchdog (WWDG) is connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced ON and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

Clock frequency measurement using TIMx

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

- Calibrating the HSI or CSI with the LSE:

  The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI or CSI. This requires to use the HSI or CSI as system clock source either directly or via PLL1. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

  The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

  The HSI and CSI oscillators have dedicated user-accessible calibration bits for this purpose (see *RCC CSI calibration register (RCC_CSICFGR)*). When HSI or CSI is used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- Calibrating the LSI with the HSI:

  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

11.4.13 RTC and TAMP clock

The RTCCLK clock source is used by RTC and TAMP, and can be either the HSE / 32, LSE or LSI clock. It is selected by programming the RTCSEL[1:0] bits in the *RCC Backup domain control register (RCC_BDCR)*. This selection cannot be modified without resetting the Backup domain. The system must always be configured so as to get a PCLK frequency greater than or equal to the RTCCLK frequency for a proper operation of the RTC. The TAMP does not require any kernel clock if only the backup registers are used, with tampers in edge detection mode. All other tamper detection modes require a kernel clock (refer to *Section 47: Tamper and backup registers (TAMP)* for more details).

The LSE is in the Backup domain, whereas the HSE and LSI clocks are not. Consequently:

- If LSE is selected as RTC and TAMP clock, these peripherals continue to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If the HSE clock divided by a prescaler is used as the RTC or TAMP clock, the RTC state is not guaranteed if the V_{DD} supply is powered off or if the internal voltage regulator is powered off (removing power from the core domain). Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.
- If the LSI is used as the RTC or TAMP clock, the RTC state is not guaranteed if the V_{DD} supply is powered off. Depending on the TAMP configuration, this one can remain functional if used in a mode that does not need any kernel clock.

When the RTC and TAMP clock is LSE or LSI, the RTC remains clocked and functional under system reset.

11.4.14 Timer clock

The timer clock frequencies are automatically defined by hardware.

There are two cases:

- If the APB prescaler equals 1, the timer clock frequencies are set to the APB domain frequency.
- Otherwise, they are set to twice (×2) the APB domain frequency.

11.4.15 Watchdog clock

If the independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced on and cannot be disabled. After the LSI oscillator temporization, the LSI 32 kHz clock is provided to the IWDG.

11.4.16 Peripherals clock gating and autonomous mode

Peripherals clock gating in Run mode

Each peripheral clock can be enabled by the corresponding EN bit in the RCC_AHBxENR and RCC_APBxENR registers.

When the peripheral clock is not active, read or write accesses to the peripheral registers are not supported.

The enable bit has a synchronization mechanism to create a glitch-free clock for the peripheral. After the enable bit is set, there the clock is active after 2 cycles of the peripheral bus clock.

Caution: Just after enabling the clock for a peripheral, the software must wait for these 2 clock cycles before accessing the peripheral registers.

Peripherals clock gating in Sleep mode

When a peripheral is enabled, its clock can be automatically gated off when the device is in Sleep mode, by clearing the peripheral LPEN bit in the RCC_AHBxLPENR and RCC_APBxLPENR registers. Both EN and LPEN bit of the peripheral must be set to keep the clock on in Sleep mode.

11.5 RCC security and privilege functional description

11.5.1 RCC TrustZone security protection modes

When the TrustZone security is activated by the TZEN option byte in the Flash option byte configuration register, the RCC is able to secure RCC configuration and status bits from being modified by non-secure accesses.

This is configured through the *RCC secure configuration register (RCC_SECCFGR)* to prevent non-secure access to read or modify the following features:

- HSE, HSE-CSS, HSI, CSI, LSI, LSE, LSE-CSS, LSCO, HSI48 configuration and status bits
- PLL1, PLL2, PLL3, AHB and APB prescalers configuration and status bits
- system clock (SYSCLK) and ICLK source clock selection and status bits
- MCO clock output configuration and STOPWUCK and STOPKERWUCK bit
- Remove reset flag RMVF configuration

If SPRIV is set in the *RCC privilege configuration register (RCC_PRIVCFGR)*, the RCC_SECCFGR register can be written only by secure and privileged access. If SPRIV is cleared in RCC_PRIVCFGR, RCC_SECCFGR can be written only by secure access, privileged or unprivileged.

RCC_SECCFGR can be read by secure, non-secure, privileged and unprivileged access.

When a peripheral is configured as secure, its related clock, reset, clock source selection and clock enable during low-power modes control bits, are also secure in the RCC_AHBxENR, RCC_APBxENR, RCC_CCIPRx and RCC_BDCR registers.

A peripheral is secure when:

- For securable peripherals by TZSC (see *Section 5.4.3: TrustZone security controller (TZSC)*), the SEC security bit corresponding to this peripheral is set in the GTZC TZSC secure configuration registers.
- For TrustZone®-aware peripherals, a security feature of this peripheral is enabled through its dedicated bits.

Table 116 summarizes the RCC secured bits following the security configuration bit in the RCC_SECCFGR register.

When one security configuration bit is set, some configuration and status bits are secured. The RCC registers may contain secure and non-secure bits:

- Secured bits: read and write operations are only allowed by a secure access. Non-secure read returns 0 and write accesses are ignored. No illegal access event is generated.
- Non-secure bits: no restriction. Read and write operations are allowed by both secure and non-secure accesses.
- A non-secure write access to RCC_SECCFGR is ignored and generates an illegal access event. An illegal access interrupt is generated if the RCC illegal access interrupt is enabled in the GTZC TZIC registers. RCC_SECCFGR can be read by secure or non-secure access.

When the TrustZone security is disabled (TZEN = 0xC3), all registers are non-secure. RCC_SECCFGR write accesses are ignored.

Table 116. RCC security configuration summary

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
HSISEC	HSION, HSIKERON, HSIRDY	RCC_CR
	HSICAL[11:0], HSITRIM[6:0]	RCC_HSICFGR
	HSIRDYIE	RCC_CIER
	HSIRDYIF	RCC_CIFR
	HSIRDYC	RCC_CICR
HSESEC	HSEON, HSERDY, HSEBYP, HSECSSON, HSEEXT	RCC_CR
	HSERDYIE, HSECSSF	RCC_CIER
	HSERDYIF, HSECSSF	RCC_CIFR
	HSERDYC, HSECSSC	RCC_CICR
CSISEC	CSISON, CSIKERON, CSISRDY	RCC_CR
	CSICAL[7:0], CSITRIM[5:0]	RCC_CSICFGR
	CSISRDYIE	RCC_CIER
	CSISRDYIF	RCC_CIFR
	CSISRDYIC	RCC_CICR
LSISEC	LSION, LSIRDY, LSIPREDIV, LSCOSEL, LSCOEN	RCC_BDCR
	LSIRDYIE	RCC_CIER
	LSIRDYIF	RCC_CIFR
	LSIRDYC	RCC_CICR
LSESEC	LSECSSON, LSECSSD, LSEDRV[1:0], LSEBYP, LSERDY, LSEON, LSEEXT, LSCOSEL, LSCOEN	RCC_BDCR
	LSERDYIE	RCC_CIER
	LSERDYF	RCC_CIFR
	LSERDYC	RCC_CICR
SYSCLKSEC	SW[1:0], SWS[1:0], STOPWUCK, STOPKERWUCK, MCO1SEL[3:0], MCO1PRE[2:0], MCO2SEL[3:0], MCO2PRE[2:0]	RCC_CFGR1
	SYSTICKSEL[1:0]	RCC_CCIPR4
	VOS[1:0]	PWR_VOSR
	TIMPRE	RCC_CFGR1
PRESCSEC	HPRE[3:0], PPRE1[2:0], PPRE2[2:0], PPRE3[2:0]	RCC_CFGR2

Table 116. RCC security configuration summary (continued)

Configuration bit in RCC_SECCFGR	Secured bits	Corresponding register
PLL1SEC	PLL1SRC[1:0], PLL1RGE[1:0], PLL1FRACEN, PLL1M[3:0], PLL1VCOSEL, PLL1PEN, PLL1QEN, PLL1REN	RCC_PLL1CFGR
	PLL1N[8:0], PLL1P[6:0], PLL1Q[6:0], PLL1R[6:0]	RCC_PLL1DIVR
	PLL1FRACN[12:0]	RCC_PLL1FRACR
	PLL1RDY, PLL1ON	RCC_CR
	PLL1RDYIE	RCC_CIER
	PLL1RDYF	RCC_CIFR
	PLL1RDYC	RCC_CICR
PLL2SEC	PLL2SRC[1:0], PLL2RGE[1:0], PLL2FRACEN, PLL2M[3:0], PLL2PEN, PLL2QEN, PLL2REN, PLL2VCOSEL	RCC_PLL2CFGR
	PLL2N[8:0], PLL2P[6:0], PLL2Q[6:0], PLL2R[6:0]	RCC_PLL2DIVR
	PLL2FRACN[12:0]	RCC_PLL2FRACR
	PLL2RDY, PLL2ON	RCC_CR
	PLL2RDYIE	RCC_CIER
	PLL2RDYF	RCC_CIFR
	PLL2RDYC	RCC_CICR
PLL3SEC	PLL3SRC[1:0], PLL3RGE[1:0], PLL3FRACEN, PLL3M[3:0], PLL3PEN, PLL3QEN, PLL3REN, PLL3VCOSEL	RCC_PLL3CFGR
	PLL3N[8:0], PLL3P[6:0], PLL3Q[6:0], PLL3R[6:0]	RCC_PLL3DIVR
	PLL3FRACN[12:0]	RCC_PLL3FRACR
	PLL3RDY, PLL3ON	RCC_CR
	PLL3RDYIE	RCC_CIER
	PLL3RDYF	RCC_CIFR
	PLL3RDYC	RCC_CICR
HSI48SEC(1)	HSI48ON, HSI48RDY	RCC_CR
	HSI48CAL[9:0]	RCC_CRRCR
	HSI48RDYE	RCC_CIER
	HSI48RDYF	RCC_CIFR
	HSI48RDYC	RCC_CICR
IPKERSECCFG	CKERPSEL[1:0]	RCC_CCIPR5
RMVFSEC	RMVF	RCC_CSR

1. TRIM field of the HSI48 is located in CRS peripheral. Be sure to secure it using CRSSEC bit in GTZC1 TZSC secure configuration register 1.

11.5.2 RCC privilege protection modes

By default, after reset, all RCC registers can be read or written with both privileged and unprivileged access except *RCC privilege configuration register (RCC_PRIVCFGR)* that can be written with privileged access only. RCC_PRIVCFGR can be read by secure and non secure, privileged and unprivileged access.

The SPRIV bit in RCC_PRIVCFGR can be written with secure privileged access only. This bit configures the privileged access of all RCC secure functions (as defined by *RCC secure configuration register (RCC_SECCFGR)* or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the SPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC secure bits is possible only with privileged access, including RCC_SECCFGR.
- The RCC secure bits can be read only with privileged access except RCC_SECCFGR and RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

The NSPRIV bit in RCC_PRIVCFGR can be written with privileged access only, secure or non-secure. This bit configures the privileged access of all RCC non-secure functions (as defined by RCC_SECCFGR, or by the GTZC for securable peripherals, or by the peripheral itself in case of TrustZone-aware peripherals).

When the NSPRIV bit is set in RCC_PRIVCFGR:

- Writing the RCC non-secure bits is possible only with privileged access.
- The RCC non-secure bits can be read only with privileged access except RCC_PRIVCFGR that can be read by privileged or unprivileged access.
- An unprivileged access to a privileged RCC bit or register is discarded: the bits are read as zero and the write to these bits is ignored (RAZ/WI).

11.6 RCC low-power modes

- AHB and APB peripheral clocks, including DMA clock, can be disabled by software.
- Sleep mode stops the CPU clock. The memory interface clocks (flash memory, caches and all SRAM interfaces) can be stopped by software during Sleep mode. The AHB to APB bridge clocks are disabled by hardware during Sleep mode when all the clocks of the peripherals connected to them are disabled.
- Stop mode stops all the clocks in the core domain and disable the PLLs, HSI, HSI48, CSI and HSE oscillators. However, HSI or CSI can be switched ON to generate a wakeup interrupt. LSI and LSE remain active in Stop mode.
- Standby mode stops all the clocks in the core domain and disable the PLLs, HSI, HSI48, CSI and HSE oscillators.

The CPU DeepSleep mode can be overridden for debugging by setting the DBG_STOP or DBG_STANDBY bit in the DBGMCU_CR register.

When exiting Stop mode, the system clock is either HSI or CSI, depending on the software configuration of STOPWUCK in the *RCC CPU domain clock configuration register 2 (RCC_CFGR2)*. The frequency (range and user trim) of the HSI is the one configured before entering Stop mode.

The other internal oscillator can be automatically woken up in addition to the one used by the system clock, to avoid waiting for the other oscillator wake-up time when the device is back in Run mode. This is done thanks to STOPKERWUCK in RCC_CFGR1.

When leaving the Standby mode, the system clock is HSI (32 MHz). The user trim is lost.

If a flash memory programming operation is ongoing, Stop or Standby mode entry is delayed until the flash memory interface access is finished. If an access to the APB domain is ongoing, Stop or Standby mode entry is delayed until the APB access is finished.

11.7 RCC interrupts

Table 117 summarizes the interrupt sources and the way to control them.

Table 117. Interrupt sources and control

Interrupt vector	Interrupt event flag	Description	Enable control bits	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby modes
RCC	LSIRDYF	LSI ready	LSIRDYIE and LSISEC = 0	Set LSIRDYC to 1	Yes	No
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 0	Set LSERDYC to 1	Yes	No
	HSIDRYF	HSI ready	HSIDRYIE and HSISEC = 0	Set HSIRDYC to 1	Yes	No
	HSERDYF	HSE ready	HSERDYIE and HSESEC = 0	Set HSERDYC to 1	Yes	No
	CSISRDYF	CSIS ready	CSISRDYIE and CSISEC = 0	Set CSISRDYC to 1	Yes	No
	HSI48RDYF	HSI48 ready	HSI48RDYIE and HSI48SEC = 0	Set HSI48RDYC to 1	Yes	No
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 0	Set PLL1RDYC to 1	Yes	No
	PLL2RDYF	PLL2 ready	PLL2RDYIE and PLL2SEC = 0	Set PLL2RDYC to 1	Yes	No
	PLL3RDYF	PLL3 ready	PLL3RDYIE and PLL3SEC = 0	Set PLL3DYC to 1	Yes	No
RCC_S[1]	LSIRDYF	LSI ready	LSIRDYIE and LSISEC = 1	Set LSIRDYC to 1	Yes	No
	LSERDYF	LSE ready	LSERDYIE and LSESEC = 1	Set LSERDYC to 1	Yes	No
	HSIDRYF	HSI ready	HSIDRYIE and HSISEC = 1	Set HSIRDYC to 1	Yes	No
	HSERDYF	HSE ready	HSERDYIE and HSESEC = 1	Set HSERDYC to 1	Yes	No
	CSISRDYF	CSIS ready	CSISRDYIE and CSISEC = 1	Set CSISRDYC to 1	Yes	No
	HSI48RDYF	HSI48 ready	HSI48RDYIE and HSI48SEC = 1	Set HSI48RDYC to 1	Yes	No
	PLL1RDYF	PLL1 ready	PLL1RDYIE and PLL1SEC = 1	Set PLL1RDYC to 1	Yes	No
	PLL2RDYF	PLL2 ready	PLL2RDYIE and PLL2SEC = 1	Set PLL2RDYC to 1	Yes	No
	PLL3RDYF	PLL3 ready	PLL3RDYIE and PLL3SEC = 1	Set PLL3RDYC to 1	Yes	No

Table 117. Interrupt sources and control (continued)

Interrupt vector	Interrupt event flag	Description	Enable control bits	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby modes
TAMP	ITAMP3F(2)	LSE CSS failure	LSECSSON and ITAMP3E(2) and ITAMP3IE(2)	Set CITAMP3F(2) to 1	Yes	Yes
NMI	HSECSSF	HSE CSS failure	(3)	Set HSECSSC to 1	Yes	No

1. The RCC secure interrupt vector is used only when TrustZone is enabled.
2. The LSE CSS failure event (LSECSSD) is connected to TAMP internal tamper 3. to get the interrupt associated to this event, the internal tamper 3 must be enabled, and the internal tamper 3 interrupt must be enabled. The ITAMP3F, ITAMP3E, ITAMP3IE, and CITAMP3F bits are in the TAMP peripheral.
3. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).

11.8 RCC registers

11.8.1 RCC clock control register (RCC_CR)

Address offset: 0x000

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PLL3 RDY	PLL3 ON	PLL2 RDY	PLL2 ON	PLL1 RDY	PLL1 ON	Res.	Res.	Res.	HSE EXT	HSE CSSON	HSE BYP	HSE RDY	HSE ON
		r	rw	r	rw	r	rw				rw	rs	rw	r	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	HSI48 RDY	HSI48 ON	Res.	CSI KERON	CSI RDY	CSI ON	Res.	Res.	HSI DIVF	HSIDIV[1:0]		HSI KERON	HSI RDY	HSI ON
		r	rw		rw	r	rw			r	rw	rw	rw	r	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **PLL3RDY:** PLL3 clock ready flag
Set by hardware to indicate that the PLL3 is locked.
0: PLL3 unlocked (default after reset)
1: PLL3 locked

Bit 28 **PLL3ON**: PLL3 enable
Set and cleared by software to enable PLL3.
Cleared by hardware when entering Stop or Standby mode.
0: PLL3 OFF (default after reset)
1: PLL3 ON

Bit 27 **PLL2RDY**: PLL2 clock ready flag
Set by hardware to indicate that the PLL is locked.
0: PLL2 unlocked
1: PLL2 locked

Bit 26 **PLL2ON**: PLL2 enable
Set and cleared by software to enable PLL2.
Cleared by hardware when entering Stop or Standby mode.
0: PLL2 OFF (default after reset)
1: PLL2 ON

Bit 25 **PLL1RDY**: PLL1 clock ready flag
Set by hardware to indicate that the PLL1 is locked.
0: PLL1 unlocked (default after reset)
1: PLL1 locked

Bit 24 **PLL1ON**: PLL1 enable
Set and cleared by software to enable PLL1.
Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to 0, if the PLL1 output is used as the system clock.
0: PLL1 OFF (default after reset)
1: PLL1 ON

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **HSEEXT**: external high speed clock type in Bypass mode

Set and reset by software to select the external clock type (analog or digital).
The external clock must be enabled with the HSEON bit to be used by the device. The HSEEXT bit can be written only if the HSE oscillator is disabled.
0: HSE in analog mode (default after reset)
1: HSE in digital mode

Bit 19 **HSECSSON**: HSE clock security system enable

Set by software to enable clock security system on HSE.
This bit is “set only” (disabled by a system reset or when the system enters in Standby mode). When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.
0: CSS on HSE OFF (clock detector OFF) (default after reset)
1: CSS on HSE ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 18 **HSEBYP**: HSE clock bypass

Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed (default after reset)
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag

Set by hardware to indicate that the HSE oscillator is stable.
0: HSE clock is not ready (default after reset)
1: HSE clock is ready

Bit 16 **HSEON**: HSE clock enable

Set and cleared by software.
Cleared by hardware to stop the HSE when entering Stop or Standby mode.
This bit cannot be cleared if the HSE is used directly (via SW mux) as system clock, or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).
0: HSE is OFF (default after reset)
1: HSE is ON

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

Bit 13 **HSI48RDY**: HSI48 clock ready flag

Set by hardware to indicate that the HSI48 oscillator is stable.
0: HSI48 clock is not ready (default after reset)
1: HSI48 clock is ready

Bit 12 **HSI48ON**: HSI48 clock enable

Set by software and cleared by software or by the hardware when the system enters to Stop or Standby mode.
0: HSI48 is OFF (default after reset)
1: HSI48 is ON

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CSIKERON**: CSI clock enable in Stop mode

Set and reset by software to force the CSI to ON, even in Stop mode, to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.
0: no effect on CSI (default after reset)
1: CSI is forced to ON even in Stop mode

Bit 9 **CSIRDY**: CSI clock ready flag

Set by hardware to indicate that the CSI oscillator is stable. This bit is activated only if the RC is enabled by CSION (it is not activated if the CSI is enabled by CSIKERON or by a peripheral request).

0: CSI clock is not ready (default after reset)

1: CSI clock is ready

Bit 8 **CSION**: CSI clock enable

Set and reset by software to enable/disable CSI clock for system and/or peripheral.

Set by hardware to force the CSI to ON when the system leaves Stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.

This bit cannot be cleared if the CSI is used directly (via SW mux) as system clock, or if the CSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).

0: CSI is OFF (default after reset)

1: CSI is ON

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **HSIDIVF**: HSI divider flag

Set and reset by hardware.

As a write operation to HSIDIV has not an immediate effect on the frequency, this flag indicates the

current status of the HSI divider. HSIDIVF goes immediately to 0 when HSIDIV value is changed, and is set back to 1 when the output frequency matches the value programmed into HSIDIV.

0: new division ratio not yet propagated to hsi_ck, hsi_ker_ck (default after reset)

1: hsi_ck, hsi_ker_ck clock frequency reflects the new HSIDIV value (default register value when the clock setting is completed).

Bits 4:3 **HSIDIV[1:0]**: HSI clock divider

Set and reset by software.

These bits allow selecting a division ratio to configure the wanted HSI clock frequency. The HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case, the new HSIDIV value is ignored.

00: division by 1, hsi_ck, hsi_ker_ck = 64 MHz

01: division by 2, hsi_ck, hsi_ker_ck = 32 MHz (default after reset)

10: division by 4, hsi_ck, hsi_ker_ck = 16 MHz

11: division by 8, hsi_ck, hsi_ker_ck = 8 MHz

Bit 2 **HSIKERON**: HSI clock enable in Stop mode

Set and reset by software to force the HSI to ON, even in Stop mode, to be quickly available as kernel clock for peripherals. This bit has no effect on the value of HSION.

0: no effect on HSI (default after reset)

1: HSI is forced to ON even in Stop mode

Bit 1 **HSIRDY**: HSI clock ready flag

Set by hardware to indicate that the HSI oscillator is stable.

0: HSI clock is not ready (default after reset)

1: HSI clock is ready

Bit 0 **HSION**: HSI clock enable

Set and cleared by software.

Set by hardware to force the HSI to ON when the product leaves Stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.

Set by hardware to force the HSI to ON when the product leaves Standby mode or in case of a failure of the HSE which is used as the system clock source.

This bit cannot be cleared if the HSI is used directly (via SW mux) as system clock, or if the HSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1).

0: HSI is OFF

1: HSI is ON (default after reset)

11.8.2 RCC HSI calibration register (RCC_HSICFGR)

Address offset: 0x010

Reset value: 0x0040 0XXX

Reset value depends on the flash memory option bytes setting.

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

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

Bits 22:16 **HSITRIM[6:0]**: HSI clock trimming

Set by software to adjust calibration.

HSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_HSI_OPT) to form the calibration trimming value.

HSICAL = HSITRIM + FLASH_HSI_OPT.

After a change of HSITRIM it takes one system clock cycle before the new HSITRIM value is updated

Note: The reset value of the field is 0x40.

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

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration

Set by hardware by option byte loading during system reset. Adjusted by software through trimming bits HSITRIM.

This field represents the sum of engineering option byte calibration and HSITRIM bits values.

11.8.3 RCC clock recovery RC register (RCC_CRRCR)

Address offset: 0x014

Reset value: 0x0000 0XXX

Reset value depends on the flash option bytes setting

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
						r	r	r	r	r	r	r	r	r	r

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

Bits 9:0 **HSI48CAL[9:0]**: Internal RC 48 MHz clock calibration
Set by hardware by option-byte loading during system reset NRESET. Read-only.

11.8.4 RCC CSI calibration register (RCC_CSICFGR)

Address offset: 0x018

Reset value: 0x0020 0XXX

Reset value depends on the flash option bytes setting

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

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

Bits 21:16 **CSITRIM[5:0]**: CSI clock trimming
Set by software to adjust calibration.
CSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_CSI_OPT) to form the calibration trimming value.
CSICAL = CSITRIM + FLASH_CSI_OPT.
Note: The reset value of the field is 0x20.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **CSICAL[7:0]**: CSI clock calibration
Set by hardware by option byte loading during system reset NRESET. Adjusted by software through trimming bits CSITRIM.
This field represents the sum of engineering option byte calibration value and CSITRIM bits value.

11.8.5 RCC clock configuration register1 (RCC_CFGR1)

Address offset: 0x01C

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 2; word, half-word and byte access

One or two wait states are inserted only if the access occurs during clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCO2SEL[2:0]			MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]				Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM PRE	Res.	RTCPRE[5:0]						STOP KER WUCK	STOP WUCK	Res.	SWS[1:0]		Res.	SW[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw		r	r		rw	rw

Bits 31:29 **MCO2SEL[2:0]**: microcontroller clock output 2

Set and cleared by software. Clock source selection may generate glitches on MCO2.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: system clock selected (**sys_ck**) (default after reset)
001: PLL2 oscillator clock selected (**pll2_p_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_p_ck**)
100: CSI clock selected (**csi_ck**)
101: LSI clock selected (**lsi_ck**)
Others: reserved

Bits 28:25 **MCO2PRE[3:0]**: MCO2 prescaler

Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 24:22 **MCO1SEL[2:0]**: Microcontroller clock output 1

Set and cleared by software. Clock source selection may generate glitches on MCO1.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
000: HSI clock selected (**hsi_ck**) (default after reset)
001: LSE oscillator clock selected (**lse_ck**)
010: HSE clock selected (**hse_ck**)
011: PLL1 clock selected (**pll1_q_ck**)
100: HSI48 clock selected (**hsi48_ck**)
Others: reserved

Bits 21:18 **MCO1PRE[3:0]**: MCO1 prescaler

Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
0000: prescaler disabled (default after reset)
0001: division by 1 (bypass)
0010: division by 2
0011: division by 3
0100: division by 4
...
1111: division by 15

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **TIMPRE**: timers clocks prescaler selection

This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domains.
0: The timers kernel clock is equal to **rcc_hclk1** if PPRE1 or PPRE2 corresponds to a division by 1 or 2, else it is equal to 2 x $\mathbf{F_{rcc_pclk1}}$ or 2 x $\mathbf{F_{rcc_pclk2}}$ (default after reset)
1: The timers kernel clock is equal to 2 x $\mathbf{F_{rcc_pclk1}}$ or 2 x $\mathbf{F_{rcc_pclk2}}$ if PPRE1 or PPRE2 corresponds to a division by 1, 2 or 4, else it is equal to 4 x $\mathbf{F_{rcc_pclk1}}$ or 4 x $\mathbf{F_{rcc_pclk2}}$

Bit 14 Reserved, must be kept at reset value.

Bits 13:8 **RTCPRE[5:0]**: HSE division factor for RTC clock

Set and cleared by software to divide the HSE to generate a clock for RTC.
Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.
000000: no clock (default after reset)
000001: no clock
000010: HSE/2
000011: HSE/3
000100: HSE/4
...
111110: HSE/62
111111: HSE/63

Bit 7 **STOPKERWUCK**: kernel clock selection after a wakeup from system Stop

Set and reset by software to select the kernel wakeup clock from system Stop.
0: HSI selected as wakeup clock from system Stop (default after reset)
1: CSI selected as wakeup clock from system Stop

Bit 6 **STOPWUCK**: system clock selection after a wakeup from system Stop

Set and reset by software to select the system wakeup clock from system Stop.
The selected clock is also used as emergency clock for the clock security system (CSS) on HSE. 0: HSI selected as wakeup clock from system Stop (default after reset)
1: CSI selected as wakeup clock from system Stop

Caution: STOPWUCK must not be modified when CSS is enabled (by HSECSSON bit) and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW =10).

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **SWS[1:0]**: system clock switch status
Set and reset by hardware to indicate which clock source is used as system clock. 000: HSI used as system clock (**hsi_ck**) (default after reset).
01: CSI used as system clock (**csi_ck**)
10: HSE used as system clock (**hse_ck**)
11: PLL1 used as system clock (**pll1_p_ck**)
Others: reserved

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SW[1:0]**: system clock and trace clock switch
Set and reset by software to select system clock and trace clock sources (**sys_ck**).
Set by hardware to force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system Stop mode, and to force the selection of the HSI in case of failure of the HSE when used directly or indirectly as system clock
00: HSI selected as system clock (**hsi_ck**) (default after reset)
01: CSI selected as system clock (**csi_ck**)
10: HSE selected as system clock (**hse_ck**)
11: PLL1 selected as system clock (**pll1_p_ck** for **sys_ck**)
Others: reserved

11.8.6 RCC CPU domain clock configuration register 2 (RCC_CFGR2)

Address offset: 0x020

Reset value: 0x0000 0000

1 or 2 wait states are inserted only if the access occurs during clock source switch.

From 0 to 15 wait states are inserted if the access occurs when the APB or AHB prescalers values update is ongoing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3 DIS	APB2 DIS	APB1 DIS	AHB4 DIS	Res.	AHB2 DIS	AHB1 DIS
									rw	rw	rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PPRE3[2:0]			Res.	PPRE2[2:0]			Res.	PPRE1[2:0]			HPRE[3:0]			
	rw	rw	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bit 22 **APB3DIS**: APB3 clock disable value.Set and cleared by software
This bit can be set to further reduce power consumption, when none of the APB3 peripherals are used and when their clocks are disabled in RCC_APB3ENR. When this bit is set, all the APB3 peripherals clocks are off.
0: APB3 clock enabled, distributed to peripherals according to their dedicated clock enable control bits
1: APB3 clock disabled

Bit 21 **APB2DIS**: APB2 clock disable value

This bit can be set to further reduce power consumption, when none of the APB2 peripherals are used and when their clocks are disabled in RCC_APB2ENR. When this bit is set, all the APB2 peripherals clocks are off.

0: APB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB2 clock disabled

Bit 20 **APB1DIS**: APB1 clock disable value

This bit can be set to further reduce power consumption, when none of the APB1 peripherals (except IWDG) are used and when their clocks are disabled in RCC_APB1ENR. When this bit is set, all the APB1 peripherals clocks are off, except for IWDG.

0: APB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: APB1 clock disabled

Bit 19 **AHB4DIS**: AHB4 clock disable

This bit can be set to further reduce power consumption, when none of the AHB4 peripherals from RCC_AHB4ENR are used and when their clocks are disabled in RCC_AHB4ENR. When this bit is set, all the AHB4 peripherals clocks from RCC_AHB4ENR are off.

0: AHB4 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB4 clock disabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **AHB2DIS**: AHB2 clock disable

This bit can be set to further reduce power consumption, when none of the AHB2 peripherals from RCC_AHB2ENR are used and when their clocks are disabled in RCC_AHB2ENR. When this bit is set, all the AHB2 peripherals clocks are off, except for SRAM2 and SRAM3.

0: AHB2 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB2 clock disabled

Bit 16 **AHB1DIS**: AHB1 clock disable

This bit can be set to further reduce power consumption, when none of the AHB1 peripherals from RCC_AHB1ENR are used and when their clocks are disabled in RCC_AHB1ENR. When this bit is set, all the AHB1 peripherals clocks are off, except for FLASH, BKPSRAM, ICACHE, DCACHE1 and SRAM1.

0: AHB1 clock enabled, distributed to peripherals according to their dedicated clock enable control bits

1: AHB1 clock disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PPRE3[2:0]**: APB low-speed prescaler (APB3)

Set and reset by software to control APB low-speed clocks division factor.

The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE3 write.

0xx: rcc_pclk3 = rcc_hclk1

100: rcc_pclk3 = rcc_hclk1 / 2

101: rcc_pclk3 = rcc_hclk1 / 4

110: rcc_pclk3 = rcc_hclk1 / 8

111: rcc_pclk3 = rcc_hclk1 / 16

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE2[2:0]**: APB high-speed prescaler (APB2)
Set and reset by software to control APB high-speed clocks division factor.
The clocks are divided with the new prescaler factor from 1 to 16 APB cycles after PPRE2 write.
0xx: rcc_pclk2 = rcc_hclk1
100: rcc_pclk2 = rcc_hclk1 / 2
101: rcc_pclk2 = rcc_hclk1 / 4
110: rcc_pclk2 = rcc_hclk1 / 8
111: rcc_pclk2 = rcc_hclk1 / 16

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE1[2:0]**: APB low-speed prescaler (APB1)
Set and reset by software to control the division factor of rcc_pclk1.
The clock is divided by the new prescaler factor from 1 to 16 cycles of rcc_hclk after PPRE write.
0xx: rcc_pclk1 = rcc_hclk1 (default after reset)
100: rcc_pclk1 = rcc_hclk1 / 2
101: rcc_pclk1 = rcc_hclk1 / 4
110: rcc_pclk1 = rcc_hclk1 / 8
111: rcc_pclk1 = rcc_hclk1 / 16

Bits 3:0 **HPRE[3:0]**: AHB prescaler
Set and reset by software to control the division factor of rcc_hclk. Changing this division ratio has an impact on the frequency of all bus matrix clocks
0xxx: rcc_hclk = sys_ck (default after reset)
1000: rcc_hclk = sys_ck / 2
1001: rcc_hclk = sys_ck / 4
1010: rcc_hclk = sys_ck / 8
1011: rcc_hclk = sys_ck / 16
1100: rcc_hclk = sys_ck / 64
1101: rcc_hclk = sys_ck / 128
1110: rcc_hclk = sys_ck / 256
1111: rcc_hclk = sys_ck / 512

Caution: Be careful when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change, and before lowering the V_{CORE} voltage, this register must be read to check that the new prescaler value has been taken into account.
Depending upon the clock source frequency and the voltage range, the software application must program a correct value in HPRE to make sure that the system frequency does not exceed the maximum frequency.

11.8.7 RCC PLL clock source selection register (RCC_PLL1CFGR)

Address offset: 0x028

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1REN	PLL1QEN	PLL1PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PLL1M[5:0]						Res.	Res.	PLL1VCOSEL	PLL1FRACEN	PLL1RGE[1:0]		PLL1SRC[1:0]	
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **PLL1REN**: PLL1 DIVR divider output enable

Set and reset by software to enable the pll1_r_ck output of the PLL1.
To save power, DIVR1EN and DIVR1 bits must be set to 0 when the pll1_r_ck is not used.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0: pll1_r_ck output disabled (default after reset)
1: pll1_r_ck output enabled

Bit 17 **PLL1QEN**: PLL1 DIVQ divider output enable

Set and reset by software to enable the pll1_q_ck output of the PLL1.
To save power, when the pll1_q_ck output of the PLL1 is not used, the pll1_q_ck must be disabled.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0: pll1_q_ck output disabled (default after reset)
1: pll1_q_ck output enabled

Bit 16 **PLL1PEN**: PLL1 DIVP divider output enable

Set and reset by software to enable the pll1_p_ck output of the PLL1.
This bit can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
To save power, when the pll1_p_ck output of the PLL1 is not used, the pll1_p_ck must be disabled.
0: pll1_p_ck output disabled (default after reset)
1: pll1_p_ck output enabled

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

Bits 13:8 **PLL1M[5:0]**: prescaler for PLL1

Set and cleared by software to configure the prescaler of the PLL1.
The hardware does not allow any modification of this prescaler when PLL1 is enabled (PLL1ON = 1 or PLL1RDY = 1).
To save power when PLL1 is not used, the value of PLL1M must be set to 0.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL1VCOSEL**: PLL1 VCO selection

Set and reset by software to select the proper VCO frequency range used for PLL1. This bit must be written before enabling the PLL1.

0: wide VCO range: 192 to 836 MHz (default after reset)

1: medium VCO range: 150 to 420 MHz

Bit 4 **PLL1FRACEN**: PLL1 fractional latch enable

Set and reset by software to latch the content of FRACN1 into the sigma-delta modulator. To latch the FRACN1 value into the sigma-delta modulator, PLL1FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN1 into the modulator.

Bits 3:2 **PLL1RGE[1:0]**: PLL1 input frequency range

Set and reset by software to select the proper reference frequency range used for PLL1. This bit must be written before enabling the PLL1.

00: PLL1 input (ref1_ck) clock range frequency between 1 and 2 MHz (default after reset)

01: PLL1 input (ref1_ck) clock range frequency between 2 and 4 MHz

10: PLL1 input (ref1_ck) clock range frequency between 4 and 8 MHz

11: PLL1 input (ref1_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL1SRC[1:0]**: PLL1M and PLLs clock source selection

Set and reset by software to select the PLL clock source. These bits can be written only when all PLLs are disabled.

To save power, when no PLL is used, the value of PLL1SRC must be set to '00'.

00: no clock send to PLL1M divider and PLLs (default after reset).

01: HSI selected as PLL clock (hsi_ck)

10: CSI selected as PLL clock (csi_ck)

11: HSE selected as PLL clock (hse_ck)

11.8.8 RCC PLL clock source selection register (RCC_PLL2CFGR)

Address offset: 0x02C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2REN	PLL2QEN	PLL2PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PLL2M[5:0]						Res.	Res.	PLL2VCOSEL	PLL2FRACEN	PLL2RGE[1:0]		PLL2SRC[1:0]	
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **PLL2REN**: PLL2 DIVR divider output enable

Set and reset by software to enable the pll2_r_ck output of the PLL2.

To save power, DIVR2EN and DIVR2 bits must be set to 0 when the pll2_r_ck is not used.

0: pll2_r_ck output disabled (default after reset)

1: pll2_r_ck output enabled

Bit 17 **PLL2QEN**: PLL2 DIVQ divider output enable
Set and reset by software to enable the pll2_q_ck output of the PLL2.
To save power, when the pll2_q_ck output of the PLL2 is not used, the pll2_q_ck must be disabled.
0: pll2_q_ck output disabled (default after reset)
1: pll2_q_ck output enabled

Bit 16 **PLL2PEN**: PLL2 DIVP divider output enable
Set and reset by software to enable the pll2_p_ck output of the PLL2.
To save power, when the pll2_p_ck output of the PLL2 is not used, the pll2_p_ck must be disabled.
0: pll2_p_ck output disabled (default after reset)
1: pll2_p_ck output enabled

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

Bits 13:8 **PLL2M[5:0]**: prescaler for PLL2
Set and cleared by software to configure the prescaler of the PLL2.
The hardware does not allow any modification of this prescaler when PLL2 is enabled (PLL2ON = 1 or PLL2RDY = 1).
To save power when PLL2 is not used, the value of PLL2M must be set to 0.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL2VCOSEL**: PLL2 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL2.
This bit must be written before enabling the PLL2.
0: wide VCO range 192 to 836 MHz (default after reset)
1: medium VCO range 150 to 420 MHz

Bit 4 **PLL2FRACEN**: PLL2 fractional latch enable
Set and reset by software to enable the pll2_p_ck output of the PLL2.
To save power, when the pll2_p_ck output of the PLL2 is not used, the pll2_p_ck must be disabled.
0: pll2_p_ck output disabled (default after reset)
1: pll2_p_ck output enabled

Bits 3:2 **PLL2RGE[1:0]**: PLL2 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL2.
These bits must be written before enabling the PLL2.
00: PLL2 input (ref2_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL2 input (ref2_ck) clock range frequency between 2 and 4 MHz
10: PLL2 input (ref2_ck) clock range frequency between 4 and 8 MHz
11: PLL2 input (ref2_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL2SRC[1:0]**: PLL2M and PLLs clock source selection
Set and reset by software to select the PLL clock source.
These bits can be written only when all PLLs are disabled.
To save power, when no PLL is used, the value of PLL2SRC must be set to '00'.
00: no clock send to PLL2M divider and PLLs (default after reset)
01: HSI selected as PLL clock (hsi_ck)
10: CSI selected as PLL clock (csi_ck)
11: HSE selected as PLL clock (hse_ck)

11.8.9 RCC PLL clock source selection register (RCC_PLL3CFGR)

Address offset: 0x030

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3REN	PLL3QEN	PLL3PEN
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PLL3M[5:0]						Res.	Res.	PLL3VCOSEL	PLL3FRACEN	PLL3RGE[1:0]		PLL3SRC[1:0]	
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **PLL3REN**: PLL3 DIVR divider output enable
Set and reset by software to enable the pll3_r_ck output of the PLL3.
To save power, DIVR2EN and DIVR2 bits must be set to 0 when the pll3_r_ck is not used.
0: pll3_r_ck output disabled (default after reset)
1: pll3_r_ck output enabled

Bit 17 **PLL3QEN**: PLL3 DIVQ divider output enable
Set and reset by software to enable the pll3_q_ck output of the PLL3.
To save power, when the pll3_q_ck output of the PLL3 is not used, the pll3_q_ck must be disabled.
0: pll3_q_ck output disabled (default after reset)
1: pll3_q_ck output enabled

Bit 16 **PLL3PEN**: PLL3 DIVP divider output enable
Set and reset by software to enable the pll3_p_ck output of the PLL3.
To save power, when the pll3_p_ck output of the PLL3 is not used, the pll3_p_ck must be disabled.
0: pll3_p_ck output disabled (default after reset)
1: pll3_p_ck output enabled

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

Bits 13:8 **PLL3M[5:0]**: prescaler for PLL3

Set and cleared by software to configure the prescaler of the PLL3.
The hardware does not allow any modification of this prescaler when PLL3 is enabled (PLL3ON = 1 or PLL3RDY = 1).
To save power when PLL3 is not used, the value of PLL3M must be set to 0.
000000: prescaler disabled (default after reset)
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32
...
111111: division by 63

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **PLL3VCOSEL**: PLL3 VCO selection

Set and reset by software to select the proper VCO frequency range used for PLL3.
This bit must be written before enabling the PLL3.
0: wide VCO range 192 to 836 MHz (default after reset)
1: medium VCO range 150 to 420 MHz

Bit 4 **PLL3FRACEN**: PLL3 fractional latch enable

Set and reset by software to latch the content of FRACN3 into the sigma-delta modulator.
To latch the FRACN3 value into the sigma-delta modulator, PLL3FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN3 into the modulator.

Bits 3:2 **PLL3RGE[1:0]**: PLL3 input frequency range

Set and reset by software to select the proper reference frequency range used for PLL3.
This bit must be written before enabling the PLL3.
00: PLL3 input (ref3_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL3 input (ref3_ck) clock range frequency between 2 and 4 MHz
10: PLL3 input (ref3_ck) clock range frequency between 4 and 8 MHz
11: PLL3 input (ref3_ck) clock range frequency between 8 and 16 MHz

Bits 1:0 **PLL3SRC[1:0]**: PLL3M and PLLs clock source selection

Set and reset by software to select the PLL clock source. These bits can be written only when all PLLs are disabled.
To save power, when no PLL is used, the value of PLL3SRC must be set to '00'.
00: no clock send to PLL3M divider and PLLs (default after reset)
01: HSI selected as PLL clock (hsi_ck)
10: CSI selected as PLL clock (csi_ck)
11: HSE selected as PLL clock (hse_ck)

11.8.10 RCC PLL1 dividers register (RCC_PLL1DIVR)

Address offset: 0x034

Reset value: 0x0101 0280

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1P[6:0]							PLL1N[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL1R[6:0]**: PLL1 DIVR division factor

Set and reset by software to control the frequency of the pll1_r_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1_r_ck = vco1_ck / 1
0000001: pll1_r_ck = vco1_ck / 2 (default after reset)
0000010: pll1_r_ck = vco1_ck / 3
0000011: pll1_r_ck = vco1_ck / 4
...
1111111: pll1_r_ck = vco1_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL1Q[6:0]**: PLL1 DIVQ division factor

Set and reset by software to control the frequency of the pll1_q_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
0000000: pll1_q_ck = vco1_ck
0000001: pll1_q_ck = vco1_ck / 2 (default after reset)
0000010: pll1_q_ck = vco1_ck / 3
0000011: pll1_q_ck = vco1_ck / 4
...
1111111: pll1_q_ck = vco1_ck / 128

Bits 15:9 **PLL1P[6:0]**: PLL1 DIVP division factor

Set and reset by software to control the frequency of the pll1_p_ck clock.
These bits can be written only when the PLL1 is disabled (PLL1ON = 0 and PLL1RDY = 0).
Note that odd division factors are not allowed.
0000000: Not allowed
0000001: pll1_p_ck = vco1_ck / 2 (default after reset)
0000010: Not allowed
0000011: pll1_p_ck = vco1_ck / 4
...
1111111: pll1_p_ck = vco1_ck / 128

Bits 8:0 **PLL1N[8:0]**: Multiplication factor for PLL1VCO

Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL1ON = 0 and PLL1RDY = 0).
0x003: PLL1N = 4
0x004: PLL1N = 5
0x005: PLL1N = 6
...
0x080: PLL1N = 129 (default after reset)
...
0x1FF: PLL1N = 512
Others: reserved

11.8.11 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x038

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL1FRACN[12:0]**: fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

Caution: The software must set correctly these bits to insure that the VCO output frequency is in the valid frequency range, that is:
* 128 to 560 MHz if PLL1VCOSEL = 0
* 150 to 420 MHz if PLL1VCOSEL = 1
VCO output frequency = F_{ref1_ck} x (PLL1N + (PLL1FRACN / 2^{13})), with
* PLL1N between 8 and 420
* PLL1FRACN between 0 and 2^{13} - 1
* The input frequency F_{ref1_ck} must be between 1 and 16 MHz.
To change the PLL1FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:
* Set the bit PLL1FRACEN to 0
* Write the new fractional value into PLL1FRACN
* Set the bit PLL1FRACEN to 1

Bits 2:0 Reserved, must be kept at reset value.

11.8.12 RCC PLL1 dividers register (RCC_PLL2DIVR)

Address offset: 0x03C

Reset value: 0x0101 0280

Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL2R[6:0]							Res.	PLL2Q[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2P[6:0]							PLL2N[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL2R[6:0]**: PLL2 DIVR division factor
Set and reset by software to control the frequency of the pll2_r_ck clock.
These bits can be written only when the PLL1 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_r_ck = vco2_ck
0000001: pll2_r_ck = vco2_ck / 2 (default after reset)
0000010: pll2_r_ck = vco2_ck / 3
0000011: pll2_r_ck = vco2_ck / 4
...
1111111: pll2_r_ck = vco2_ck / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL2Q[6:0]**: PLL2 DIVQ division factor
Set and reset by software to control the frequency of the pll2_q_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_q_ck = vco2_ck
0000001: pll2_q_ck = vco2_ck / 2 (default after reset)
0000010: pll2_q_ck = vco2_ck / 3
0000011: pll2_q_ck = vco2_ck / 4
...
1111111: pll2_q_ck = vco2_ck / 128

Bits 15:9 **PLL2P[6:0]**: PLL2 DIVP division factor
Set and reset by software to control the frequency of the pll2_p_ck clock.
These bits can be written only when the PLL2 is disabled (PLL2ON = 0 and PLL2RDY = 0).
0000000: pll2_p_ck = vco2_ck
0000001: pll2_p_ck = vco2_ck / 2 (default after reset)
0000010: pll2_p_ck = vco2_ck / 3
0000011: pll2_p_ck = vco2_ck / 4
...
1111111: pll2_p_ck = vco2_ck / 128

Bits 8:0 **PLL2N[8:0]**: Multiplication factor for PLL2VCO
Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL2ON = 0 and PLL2RDY = 0).
0x003: PLL2N = 4
0x004: PLL2N = 5
0x005: PLL2N = 6
...
0x080: PLL2N = 129 (default after reset)
...
0x1FF: PLL2N = 512
Others: reserved

11.8.13 RCC PLL2 fractional divider register (RCC_PLL2FRACR)

Address offset: 0x040

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL2FRACN[12:0]:** fractional part of the multiplication factor for PLL2 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:
* 128 to 560 MHz if PLL2VCOSEL = 0
* 150 to 420 MHz if PLL2VCOSEL = 1
VCO output frequency = Fref2_ck x (PLL2N + (PLL2FRACN / 213)), with
* PLL2N between 8 and 420
* PLL2FRACN can be between 0 and 213- 1
* The input frequency Fref2_ck must be between 1 and 16 MHz.
To change the PLL2FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:
* Set the bit PLL2FRACEN to 0
* Write the new fractional value into PLL2FRACN
* Set the bit PLL2FRACEN to 1

Bits 2:0 Reserved, must be kept at reset value.

11.8.14 RCC PLL3 dividers register (RCC_PLL3DIVR)

Address offset: 0x044

Reset value: 0x0101 0280

Access: no wait state; word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL3R[6:0]							Res.	PLL3Q[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL3P[6:0]							PLL3N[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **PLL3R[6:0]**: PLL3 DIVR division factor
Set and reset by software to control the frequency of the **pll3_r_ck** clock.
These bits can be written only when the PLL1 is disabled (PLL3ON = 0 and PLL3RDY = 0).
0000000: **pll3_r_ck** = **vco3_ck**
0000001: **pll3_r_ck** = **vco3_ck** / 2 (default after reset)
0000010: **pll3_r_ck** = **vco3_ck** / 3
0000011: **pll3_r_ck** = **vco3_ck** / 4
...
1111111: **pll3_r_ck** = **vco3_ck** / 128

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **PLL3Q[6:0]**: PLL3 DIVQ division factor
Set and reset by software to control the frequency of the pll3_q_ck clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).
0000000: pll3_q_ck = vco3_ck
0000001: pll3_q_ck = vco3_ck / 2 (default after reset)
0000010: pll3_q_ck = vco3_ck / 3
0000011: pll3_q_ck = vco3_ck / 4
...
1111111: pll3_q_ck = vco3_ck / 128

Bits 15:9 **PLL3P[6:0]**: PLL3 DIVP division factor
Set and reset by software to control the frequency of the pll3_p_ck clock.
These bits can be written only when the PLL3 is disabled (PLL3ON = 0 and PLL3RDY = 0).
0000000: pll3_p_ck = vco3_ck
0000001: pll3_p_ck = vco3_ck / 2 (default after reset)
0000010: pll3_p_ck = vco3_ck / 3
0000011: pll3_p_ck = vco3_ck / 4
...
1111111: pll3_p_ck = vco3_ck / 128

Bits 8:0 **PLL3N[8:0]**: Multiplication factor for PLL3VCO
Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL3ON = 0 and PLL3RDY = 0).
0x003: PLL3N = 4
0x004: PLL3N = 5
0x005: PLL3N = 6
...
0x080: PLL3N = 129 (default after reset)
...
0x1FF: PLL3N = 512
Others: reserved

11.8.15 RCC PLL3 fractional divider register (RCC_PLL3FRACR)

Address offset: 0x048

Reset value: 0x0000 0000

Access: no wait state; word and half-word access

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL3FRACN[12:0]													Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:3 **PLL3FRACN[12:0]**: fractional part of the multiplication factor for PLL3 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL3 VCO.

Caution: The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:
* 128 to 560 MHz if PLL3VCOSEL = 0
* 150 to 420 MHz if PLL3VCOSEL = 1
VCO output frequency = Fref3_ck x (PLL3N + (PLL3FRACN / 213)), with
* PLL3N between 8 and 420
* PLL3FRACN can be between 0 and 213 - 1
* The input frequency Fref3_ck must be between 1 and 16 MHz.
To change the PLL3FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:
* Set the bit PLL3FRACEN to 0
* Write the new fractional value into PLL3FRACN
* Set the bit PLL3FRACEN to 1

Bits 2:0 Reserved, must be kept at reset value.

11.8.16 RCC clock source interrupt enable register (RCC_CIER)

Address offset: 0x050

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	CSIRDYIE	LSERDYIE	LSIRDYIE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PLL3RDYIE**: PLL3 ready interrupt enable

Set and reset by software to enable/disable interrupt caused by PLL3 lock.
0: PLL3 lock interrupt disabled (default after reset)
1: PLL3 lock interrupt enabled

Bit 7 **PLL2RDYIE**: PLL2 ready interrupt enable

Set and reset by software to enable/disable interrupt caused by PLL2 lock.
0: PLL2 lock interrupt disabled (default after reset)
1: PLL2 lock interrupt enabled

Bit 6 **PLL1RDYIE**: PLL1 ready interrupt enable
Set and reset by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled (default after reset)
1: PLL1 lock interrupt enabled

Bit 5 **HSI48RDYIE: HSI48 ready interrupt enable**
Set and reset by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.
0: HSI48 ready interrupt disabled (default after reset)
1: HSI48 ready interrupt enabled

Bit 4 **HSERDYIE**: HSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 2 **CSIRDYIE**: CSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.
0: CSI ready interrupt disabled (default after reset)
1: CSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
Set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

11.8.17 RCC clock source interrupt flag register (RCC_CIFR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSF	Res.	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	CSIRDYF	LSERDYF	LSIRDYF
					r		r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSF**: HSE clock security system interrupt flag
Reset by software by writing HSECSSC bit.
Set by hardware in case of HSE clock failure.
0: no clock security interrupt caused by HSE clock failure (default after reset)
1: clock security interrupt caused by HSE clock failure

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PLL3RDYF**: PLL3 ready interrupt flag
Reset by software by writing PLL3RDYC bit.
Set by hardware when the PLL3 locks and PLL3RDYIE is set.
0: no clock ready interrupt caused by PLL3 lock (default after reset)
1: clock ready interrupt caused by PLL3 lock

Bit 7 **PLL2RDYF**: PLL2 ready interrupt flag
Reset by software by writing PLL2RDYC bit.
Set by hardware when the PLL2 locks and PLL2RDYIE is set.
0: no clock ready interrupt caused by PLL2 lock (default after reset)
1: clock ready interrupt caused by PLL2 lock

Bit 6 **PLL1RDYF**: PLL1 ready interrupt flag
Reset by software by writing PLL1RDYC bit.
Set by hardware when the PLL1 locks and PLL1RDYIE is set.
0: no clock ready interrupt caused by PLL1 lock (default after reset)
1: clock ready interrupt caused by PLL1 lock

Bit 5 **HSI48RDYF**: HSI48 ready interrupt flag
Reset by software by writing HSI48RDYC bit.
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
0: no clock ready interrupt caused by the HSI48 oscillator (default after reset)
1: clock ready interrupt caused by the HSI48 oscillator

Bit 4 **HSERDYF**: HSE ready interrupt flag
Reset by software by writing HSERDYC bit.
Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: no clock ready interrupt caused by the HSE (default after reset)
1: clock ready interrupt caused by the HSE

Bit 3 **HSIRDYF**: HSI ready interrupt flag
Reset by software by writing HSIRDYC bit.
Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
0: no clock ready interrupt caused by the HSI (default after reset)
1: clock ready interrupt caused by the HSI

Bit 2 **CSIRDYF**: CSI ready interrupt flag
Reset by software by writing CSIRDYC bit.
Set by hardware when the CSI clock becomes stable and CSIRDYIE is set.
0: no clock ready interrupt caused by the CSI (default after reset)
1: clock ready interrupt caused by the CSI

Bit 1 **LSERDYF**: LSE ready interrupt flag
Reset by software by writing LSERDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: no clock ready interrupt caused by the LSE (default after reset)
1: clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF**: LSI ready interrupt flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
0: no clock ready interrupt caused by the LSI (default after reset)
1: clock ready interrupt caused by the LSI

11.8.18 RCC clock source interrupt clear register (RCC_CICR)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HSECSSC	Res.	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	CSIRDYC	LSERDYC	LSIRDYC
					rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **HSECSSC**: HSE clock security system interrupt clear
Set by software to clear HSECSSF.
Reset by hardware when clear done.
0: HSECSSF no effect (default after reset)
1: HSECSSF cleared

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PLL3RDYC**: PLL3 ready interrupt clear
Set by software to clear PLL3RDYF.
Reset by hardware when clear done.
0: PLL3RDYF no effect (default after reset)
1: PLL3RDYF cleared

Bit 7 **PLL2RDYC**: PLL2 ready interrupt clear
Set by software to clear PLL2RDYF.
Reset by hardware when clear done.
0: PLL2RDYF no effect (default after reset)
1: PLL2RDYF cleared

Bit 6 **PLL1RDYC**: PLL1 ready interrupt clear
Set by software to clear PLL1RDYF.
Reset by hardware when clear done.
0: PLL1RDYF no effect (default after reset)
1: PLL1RDYF cleared

Bit 5 **HSI48RDYC**: HSI48 ready interrupt clear
Set by software to clear HSI48RDYF.
Reset by hardware when clear done.
0: HSI48RDYF no effect (default after reset)
1: HSI48RDYF cleared

Bit 4 **HSERDYC**: HSE ready interrupt clear
Set by software to clear HSERDYF.
Reset by hardware when clear done.
0: HSERDYF no effect (default after reset)
1: HSERDYF cleared

Bit 3 **HSIRDYC**: HSI ready interrupt clear
Set by software to clear HSIRDYF.
Reset by hardware when clear done.
0: HSIRDYF no effect (default after reset)
1: HSIRDYF cleared

Bit 2 **CSIRDYC**: HSI ready interrupt clear
Set by software to clear CSIRDYF.
Reset by hardware when clear done.
0: CSIRDYF no effect (default after reset)
1: CSIRDYF cleared

Bit 1 **LSERDYC**: LSE ready interrupt clear
Set by software to clear LSERDYF.
Reset by hardware when clear done.
0: LSERDYF no effect (default after reset)
1: LSERDYF cleared

Bit 0 **LSIRDYC**: LSI ready interrupt clear
Set by software to clear LSIRDYF.
Reset by hardware when clear done.
0: LSIRDYF no effect (default after reset)
1: LSIRDYF cleared

11.8.19 RCC AHB1 reset register (RCC_AHB1RSTR)

Address offset: 0x060

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMAC RST	CORDIC RST	Res.	CRC RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2 RST	GPDMA1 RST
rw	rw		rw											rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **ETHRST**: ETHRST block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset ETHRST block (default after reset)
1: resets the ETHRST block

Bit 18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGRST**: RAMCFG block reset
Set and reset by software.
0: does not reset RAMCFG block (default after reset)
1: resets RAMCFG block

Bit 16 Reserved, must be kept at reset value.

Bit 15 **FMACRST**: FMAC block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset FMAC block (default after reset)
1: resets FMAC block

Bit 14 **CORDICRST**: CORDIC block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset CORDIC block (default after reset)
1: resets CORDIC block

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCRST**: CRC block reset
Set and reset by software.
0: does not reset CRC block (default after reset)
1: resets CRC block

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **GPDMA2RST**: GPDMA2 block reset
Set and reset by software.
0: does not reset GPDMA2 block (default after reset)
1: resets GPDMA2 block

Bit 0 **GPDMA1RST**: GPDMA1 block reset
Set and reset by software.
0: does not reset GPDMA1 block (default after reset)
1: resets GPDMA1 block

11.8.20 RCC AHB2 peripheral reset register (RCC_AHB2RSTR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAES RST	PKA RST	RNG RST	HASH RST	AES RST
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DCMI_ PSSI RST	DAC1 RST	ADC RST	Res.	GPIOI RST	GPIOH RST	GPIOG RST	GPIOF RST	GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST
			rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **SAESRST**: SAES block reset
Set and reset by software.
0: does not reset SAES block (default after reset)
1: resets SAES block

Bit 19 **PKARST**: PKA block reset
Set and reset by software.
0: does not reset PKA block (default after reset)
1: resets PKA block

Bit 18 **RNGRST**: RNG block reset
Set and reset by software.
0: does not reset RNG block (default after reset)
1: resets RNG block

Bit 17 **HASHRST**: HASH block reset
Set and reset by software.
0: does not reset HASH block (default after reset)
1: resets HASH block

Bit 16 **AESRST**: AES block reset
Set and reset by software.
0: does not reset AES block (default after reset)
1: resets AES block

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSIRST**: digital camera interface block reset (DCMI or PSSI depending which interface is active)
Set and reset by software.
0: does not reset the DCMI/PSSI block (default after reset)
1: resets the DCMI/PSSI block

Bit 11 **DAC1RST**: DAC block reset
Set and reset by software.
0: does not reset DAC block (default after reset)
1: resets DAC block

Bit 10 **ADCRST**: ADC1 and 2 blocks reset
Set and reset by software.
0: does not reset ADC1 and 2 blocks (default after reset)
1: resets ADC1 and 2 blocks

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOIRST**: GPIOI block reset
Set and reset by software.Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the GPIOI block (default after reset)
1: resets the GPIOI block

Bit 7 **GPIOHRST**: GPIOH block reset
Set and reset by software.
0: does not reset the GPIOH block (default after reset)
1: resets the GPIOH block

Bit 6 **GPIOGRST**: GPIOG block reset
Set and reset by software.
0: does not reset the GPIOG block (default after reset)
1: resets the GPIOG block

Bit 5 **GPIOFRST**: GPIOF block reset
Set and reset by software.
0: does not reset the GPIOF block (default after reset)
1: resets the GPIOF block

Bit 4 **GPIOERST**: GPIOE block reset
Set and reset by software.
0: does not reset the GPIOE block (default after reset)
1: resets the GPIOE block

Bit 3 **GPIODRST**: GPIOD block reset
Set and reset by software.
0: does not reset the GPIOD block (default after reset)
1: resets the GPIOD block

Bit 2 **GPIOCRST**: GPIOC block reset
Set and reset by software.
0: does not reset the GPIOC block (default after reset)
1: resets the GPIOC block

Bit 1 **GPIOBRST**: GPIOB block reset
Set and reset by software.
0: does not reset the GPIOB block (default after reset)
1: resets the GPIOB block

Bit 0 **GPIOARST**: GPIOA block reset
Set and reset by software.
0: does not reset the GPIOA block (default after reset)
1: resets the GPIOA block

11.8.21 RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1 RST	Res.	Res.	Res.	FMC RST
											rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SDMMC2 RST	SDMMC1 RST	Res.	Res.	Res.	OTFDEC1 RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw				rw							

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1RST**: OCTOSPI1 block reset
Set and reset by software.
0: does not reset OCTOSPI1 block (default after reset)
1: resets OCTOSPI1 block

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **FMCRST**: FMC block reset
Set and reset by software.
0: does not reset FMC block (default after reset)
1: resets FMC block

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **SDMMC2RST**: SDMMC2 and SDMMC2 delay blocks reset
Set and reset by software.Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset SDMMC2 and SDMMC2 delay blocks (default after reset)
1: resets SDMMC2 and SDMMC2 delay blocks

Bit 11 **SDMMC1RST**: SDMMC1 and SDMMC1 delay blocks reset
Set and reset by software.
0: does not reset SDMMC1 and SDMMC1 delay blocks (default after reset)
1: resets SDMMC1 and SDMMC1 delay blocks

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **OTFDEC1RST**: OTFDEC1 block reset
Set and reset by software.
0: does not reset OTFDEC1 block (default after reset)
1: resets OTFDEC1 block

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

11.8.22 RCC APB1 peripheral low reset register (RCC_APB1LRSTR)

Address offset: 0x074

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 RST	UART7 RST	Res.	CEC RST	USART11 RST	USART10 RST	USART6 RST	CRS RST	I3C1 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	Res.
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Res.	Res.	Res.	Res.	Res.	TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8RST**: UART8 block reset
Set and reset by software.Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the UART8 block (default after reset)
1: resets the UART8 block

Bit 30 **UART7RST**: UART7 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the UART7 block (default after reset)
1: resets the UART7 block

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CECRST**: HDMI-CEC block reset
Set and reset by software.
0: does not reset the HDMI-CEC block (default after reset)
1: resets the HDMI-CEC block

Bit 27 **USART11RST**: USART11 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the USART11 block (default after reset)
1: resets the USART11 block

Bit 26 **USART10RST**: USART10 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the USART10 block (default after reset)
1: resets the USART10 block

Bit 25 **USART6RST**: USART6 block reset
Set and reset by software.
0: does not reset the USART6 block (default after reset)
1: resets the USART6 block

Bit 24 **CRSRST**: CRS block reset
Set and reset by software.
0: does not reset the CRS block (default after reset)
1: resets the CRS block

Bit 23 **I3C1RST**: I3C1 block reset
Set and reset by software.
0: does not reset the I3C1 block (default after reset)
1: resets the I3C1 block

Bit 22 **I2C2RST**: I2C2 block reset
Set and reset by software.
0: does not reset the I2C2 block (default after reset)
1: resets the I2C2 block

Bit 21 **I2C1RST**: I2C1 block reset
Set and reset by software.
0: does not reset the I2C1 block (default after reset)
1: resets the I2C1 block

Bit 20 **UART5RST**: UART5 block reset
Set and reset by software.
0: does not reset the UART5 block (default after reset)
1: resets the UART5 block

Bit 19 **UART4RST**: UART4 block reset
Set and reset by software.
0: does not reset the UART4 block (default after reset)
1: resets the UART4 block

Bit 18 **USART3RST**: USART3 block reset
Set and reset by software.
0: does not reset the USART3 block (default after reset)
1: resets the USART3 block

Bit 17 **USART2RST**: USART2 block reset
Set and reset by software.
0: does not reset the USART2 block (default after reset)
1: resets the USART2 block

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3RST**: SPI3 block reset
Set and reset by software.
0: does not reset the SPI3 block (default after reset)
1: resets the SPI3 block

Bit 14 **SPI2RST**: SPI2 block reset
Set and reset by software.
0: does not reset the SPI2 block (default after reset)
1: resets the SPI2 block

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **TIM14RST**: TIM14 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the TIM14 block (default after reset)
1: resets the TIM14 block

Bit 7 **TIM13RST**: TIM13 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the TIM13 block (default after reset)
1: resets the TIM13 block

Bit 6 **TIM12RST**: TIM12 block reset
Set and reset by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block

Bit 5 **TIM7RST**: TIM7 block reset
Set and reset by software.
0: does not reset the TIM7 block (default after reset)
1: resets the TIM7 block

Bit 4 **TIM6RST**: TIM6 block reset
Set and reset by software.
0: does not reset the TIM6 block (default after reset)
1: resets the TIM6 block

Bit 3 **TIM5RST**: TIM5 block reset
Set and reset by software.
0: does not reset the TIM5 block (default after reset)
1: resets the TIM5 block

Bit 2 **TIM4RST**: TIM4 block reset
Set and reset by software.
0: does not reset the TIM4 block (default after reset)
1: resets the TIM4 block

Bit 1 **TIM3RST**: TIM3 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the TIM3 block (default after reset)
1: resets the TIM3 block

Bit 0 **TIM2RST**: TIM2 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the TIM2 block (default after reset)
1: resets the TIM2 block

11.8.23 RCC APB1 peripheral high reset register (RCC_APB1HRSTR)

Address offset: 0x078

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	LPTIM2RST	Res.	DTSRST	Res.	UART12RST	UART9RST
						rw				rw		rw		rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UCPD1RST**: UCPD1 block reset
Set and reset by software.
0: does not reset the UCPD block (default after reset)
1: resets the UCPD block

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCANRST**: FDCAN1 and FDCAN2 blocks reset
Set and reset by software.
0: does not reset the FDCAN1 and FDCAN2 blocks (default after reset)
1: resets the FDCAN1 and FDCAN2 blocks

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2RST**: LPTIM2 block reset
Set and reset by software.
0: does not reset the LPTIM2 block (default after reset)
1: resets the LPTIM2 block

Bit 4 Reserved, must be kept at reset value.

Bit 3 **DTSRST**: DTS block reset
Set and reset by software.
0: does not reset the DTS block (default after reset)
1: resets the DTS block

Bit 2 Reserved, must be kept at reset value.

Bit 1 **UART12RST**: UART12 block reset
Set and reset by software.
0: does not reset the UART12 block (default after reset)
1: resets the UART12 block

Bit 0 **UART9RST**: UART9 block reset
Set and reset by software.
0: does not reset UART9 block (default after reset)
1: resets UART9 block

11.8.24 RCC APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x07C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB RST	Res.	SAI2 RST	SAI1 RST	SPI6 RST	SPI4 RST	TIM17 RST	TIM16 RST	TIM15 RST
							rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	TIM8 RST	SPI1 RST	TIM1 RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBRST**: USB block reset
Set and reset by software.
0: does not reset the USB block (default after reset)
1: resets the USB block

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2RST**: SAI2 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the SAI2 block (default after reset)
1: resets the SAI2 block

Bit 21 **SAI1RST**: SAI1 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the SAI1 (default after reset)
1: resets the SAI1

Bit 20 **SPI6RST**: SPI6 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the SPI6 block (default after reset)
1: resets the SPI6 block

Bit 19 **SPI4RST**: SPI4 block reset
Set and reset by software.
0: does not reset the SPI4 block (default after reset)
1: resets the SPI4 block

Bit 18 **TIM17RST**: TIM17 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the TIM17 block (default after reset)
1: resets the TIM17 block

Bit 17 **TIM16RST**: TIM16 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the TIM16 block (default after reset)
1: resets the TIM16 block

Bit 16 **TIM15RST**: TIM15 block reset
Set and reset by software.
0: does not reset the TIM15 block (default after reset)
1: resets the TIM15 block

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1RST**: USART1 block reset
Set and reset by software.
0: does not reset the USART1 block (default after reset)
1: resets the USART1 block

Bit 13 **TIM8RST**: TIM8 block reset
Set and reset by software.
0: does not reset the TIM8 block (default after reset)
1: resets the TIM8 block

Bit 12 **SPI1RST**: SPI1 block reset
Set and reset by software.
0: does not reset the SPI1 block (default after reset)
1: resets the SPI1 block

Bit 11 **TIM1RST**: TIM1 block reset
Set and reset by software.
0: does not reset the TIM1 block (default after reset)
1: resets the TIM1 block

Bits 10:0 Reserved, must be kept at reset value.

11.8.25 RCC APB3 peripheral reset register (RCC_APB3RSTR)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREF RST	Res.	Res.	Res.	Res.
											rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM6 RST	LPTIM5 RST	LPTIM4 RST	LPTIM3 RST	LPTIM1 RST	Res.	I3C2 RST	I2C4 RST	I2C3 RST	LPUART1 RST	SPI5 RST	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw					

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VREFRST**: VREFBUF block reset
Set and reset by software.
0: does not reset the VREFBUF block (default after reset)
1: resets the VREFBUF block

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **LPTIM6RST**: LPTIM6 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the LPTIM6 block (default after reset)
1: resets the LPTIM6 block

Bit 14 **LPTIM5RST**: LPTIM5 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the LPTIM5 block (default after reset)
1: resets the LPTIM5 block

Bit 13 **LPTIM4RST**: LPTIM4 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the LPTIM4 block (default after reset)
1: resets the LPTIM4 block

Bit 12 **LPTIM3RST**: LPTIM3 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the LPTIM3 block (default after reset)
1: resets the LPTIM3 block

Bit 11 **LPTIM1RST**: LPTIM1 block reset
Set and reset by software.
0: does not reset the LPTIM1 block (default after reset)
1: resets the LPTIM1 block

Bit 10 Reserved, must be kept at reset value.

Bit 9 **I3C2RST**: I3C2 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the I3C2 block (default after reset)
1: resets the I3C2 block

Bit 8 **I2C4RST**: I2C4 block reset
Set and reset by software.
0: does not reset the I2C4 block (default after reset)
1: resets the I2C4 block

Bit 7 **I2C3RST**: I2C3 block reset
Set and reset by software.
0: does not reset the I2C3 block (default after reset)
1: resets the I2C3 block

Bit 6 **LPUART1RST**: LPUART1 block reset
Set and reset by software.
0: does not reset the LPUART1 block (default after reset)
1: resets the LPUART1 block

Bit 5 **SPI5RST**: SPI5 block reset
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: does not reset the SPI5 block (default after reset)
1: resets the SPI5 block

Bits 4:0 Reserved, must be kept at reset value.

11.8.26 RCC AHB1 peripherals clock register (RCC_AHB1ENR)

Address offset: 0x088

Reset value: 0xD000 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1EN	DCACHEEN	Res.	BKPRAMEN	Res.	Res.	Res.	TZSC1EN	Res.	Res.	ETHRXEN	ETHTXEN	ETHEN	Res.	RAMCFGEN	Res.
rw	rw		rw				rw			rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMACEN	CORDICEN	Res.	CRCEN	Res.	Res.	Res.	FLITFEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2EN	GPDMA1EN
rw	rw		rw				rw							rw	rw

Bit 31 **SRAM1EN**:SRAM1 clock enable
Set and reset by software.
0: SRAM1 clock disabled
1: SRAM1 clock enabled (default after reset)

Bit 30 **DCACHEEN**: DCACHE clock enable
Set and reset by software
0: DCACHE peripheral clock disabled (default after reset)
1: DCACHE peripheral clock enabled

Bit 29 Reserved, must be kept at reset value.

Bit 28 **BKPRAMEN**: BKPRAM clock enable
Set and reset by software.
0: BKPRAM peripheral clock disabled (default after reset)
1: BKPRAM peripheral clock enabled

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TZSC1EN**: TZSC1 clock enable
Set and reset by software.
0: TZSC1 peripheral clock disabled (default after reset)
1: TZSC1 peripheral clock enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ETHRXEN**: ETHRX clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: ETHRX clock disabled (default after reset)
1: ETHRX clock enabled

Bit 20 **ETHTXEN**: ETHTX clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: ETHTX clock disabled (default after reset)
1: ETHTX clock enabled

Bit 19 **ETHEN**: ETH clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: ETH peripheral clock disabled (default after reset)
1: ETH peripheral clock enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGEN**: RAMCFG clock enable
Set and reset by software.
0: RAMCFG peripheral clock disabled (default after reset)
1: RAMCFG peripheral clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **FMACEN**: FMAC clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: FMAC peripheral clock disabled (default after reset)
1: FMAC peripheral clock enabled

Bit 14 **CORDICEN**: CORDIC clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: CORDIC peripheral clock disabled (default after reset)
1: CORDIC peripheral clock enabled

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCEN**: CRC clock enable
Set and reset by software.
0: CRC peripheral clock disabled (default after reset)
1: CRC peripheral clock enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLITFEN**: Flash interface clock enable
Set and reset by software.
0: FLASH interface clock disabled
1: FLASH interface clock enabled (default after reset)

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

Bit 1 **GPDMA2EN**: GPDMA2 clock enable
Set and reset by software.
0: GPDMA2 peripheral clock disabled (default after reset)
1: GPDMA2 peripheral clock enabled

Bit 0 **GPDMA1EN**: GPDMA1 clock enable
Set and reset by software.
0: GPDMA1 peripheral clock disabled (default after reset)
1: GPDMA1 peripheral clock enabled

11.8.27 RCC AHB2 peripheral clock register (RCC_AHB2ENR)

Address offset: 0x08C

Reset value: 0xC000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM3EN	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAESEN	PKAEN	RNGEN	HASHEN	AESEN
rw	rw										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DCMI_PSSIEN	DAC1EN	ADCEN	Res.	GPIOIEN	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
			rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SRAM3EN**: SRAM3 clock enable
Set and reset by software.
0: SRAM3 clock disabled
1: SRAM3 clock enabled (default after reset)

Bit 30 **SRAM2EN**: SRAM2 clock enable
Set and reset by software.
0: SRAM2 clock disabled
1: SRAM2 clock enabled (default after reset)

Bits 29:21 Reserved, must be kept at reset value.

Bit 20 **SAESEN**: SAES clock enable
Set and reset by software.
0: SAES peripheral clock disabled (default after reset)
1: SAES peripheral clock enabled

Bit 19 **PKAEN**: PKA clock enable
Set and reset by software.
0: PKA peripheral clock disabled (default after reset)
1: PKA peripheral clock enabled

Bit 18 **RNGEN**: RNG clock enable
Set and reset by software.
0: RNG peripheral clock disabled (default after reset)
1: RNG peripheral clock enabled

Bit 17 **HASHEN**: HASH clock enable
Set and reset by software.
0: HASH peripheral clock disabled (default after reset)
1: HASH peripheral clock enabled

Bit 16 **AESEN**: AES clock enable
Set and reset by software.
0: AES peripheral clock disabled (default after reset)
1: AES peripheral clock enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSIEN**: digital camera interface clock enable (DCMI or PSSI, depending which interface is active)
Set and reset by software.
0: DCMI/PSSI peripheral clock disabled (default after reset)
1: DCMI/PSSI peripheral clock enabled

Bit 11 **DAC1EN**: DAC clock enable
Set and reset by software.
0: DAC peripheral clock disabled (default after reset)
1: DAC peripheral clock enabled

Bit 10 **ADCEN**: ADC1 and 2 peripherals clock enable
Set and reset by software.
0: ADC1 and 2 peripherals clock disabled (default after reset)
1: ADC1 and 2 peripherals clock enabled

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOIEN**: GPIOI clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: GPIOI peripheral clock disabled (default after reset)
1: GPIOI peripheral clock enabled

Bit 7 **GPIOHEN**: GPIOH clock enable
Set and reset by software.
0: GPIOH peripheral clock disabled (default after reset)
1: GPIOH peripheral clock enabled

Bit 6 **GPIOGEN**: GPIOG clock enable
Set and reset by software.
0: GPIOG peripheral clock disabled (default after reset)
1: GPIOG peripheral clock enabled

Bit 5 **GPIOFEN**: GPIOF clock enable
Set and reset by software.
0: GPIOF peripheral clock disabled (default after reset)
1: GPIOF peripheral clock enabled

Bit 4 **GPIOEEN**: GPIOE clock enable
Set and reset by software.
0: GPIOE peripheral clock disabled (default after reset)
1: GPIOE peripheral clock enabled

Bit 3 **GPIODEN**: GPIOD clock enable
Set and reset by software.
0: GPIOD peripheral clock disabled (default after reset)
1: GPIOD peripheral clock enabled

Bit 2 **GPIOCEN**: GPIOC clock enable
Set and reset by software.
0: GPIOC peripheral clock disabled (default after reset)
1: GPIOC peripheral clock enabled

Bit 1 **GPIOBEN**: GPIOB clock enable
Set and reset by software.
0: GPIOB peripheral clock disabled (default after reset)
1: GPIOB peripheral clock enabled

Bit 0 **GPIOAEN**: GPIOA clock enable
Set and reset by software.
0: GPIOA peripheral clock disabled (default after reset)
1: GPIOA peripheral clock enabled

11.8.28 RCC AHB4 peripheral clock register (RCC_AHB4ENR)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1 EN	Res.	Res.	Res.	FMC EN
											rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SDMMC2 EN	SDMMC1 EN	Res.	Res.	Res.	OTFDEC1 EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw				rw							

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1EN**: OCTOSPI1 clock enable
Set and reset by software.
0: OCTOSPI1 peripheral clock disabled (default after reset)
1: OCTOSPI1 peripheral clock enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **FMCEN**: FMC clock enable
Set and reset by software.
0: FMC peripheral clock disabled (default after reset)
1: FMC peripheral clock enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **SDMMC2EN**: SDMMC2 and SDMMC2 delay peripheral clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SDMMC2 and SDMMC2 delay peripherals clock disabled (default after reset)
1: SDMMC2 and SDMMC2 delay peripherals clock enabled

Bit 11 **SDMMC1EN**: SDMMC1 and SDMMC1 delay peripheral clock enable reset
0: SDMMC1 and SDMMC1 delay peripherals clock disabled (default after reset)
1: SDMMC1 and SDMMC1 delay peripherals clock enabled

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **OTFDEC1EN**: OTFDEC1 clock enable
Set and reset by software.
0: OTFDEC1 peripheral clock disabled (default after reset)
1: OTFDEC1 peripheral clock enabled

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

11.8.29 RCC APB1 peripheral clock register (RCC_APB1LENR)

Address offset: 0x09C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 EN	UART7 EN	Res.	CEC EN	USART11 EN	USART10 EN	USART6 EN	CRS EN	I3C1 EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	Res.
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Res.	Res.	WWDG EN	Res.	Res.	TIM14 EN	TIM13 EN	TIM12 EN	TIM7 EN	TIM6 EN	TIM5 EN	TIM4 EN	TIM3 EN	TIM2 EN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8EN**: UART8 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: UART8 peripheral clock disabled (default after reset)
1: UART8 peripheral clock enabled

Bit 30 **UART7EN**: UART7 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: UART7 peripheral clock disabled (default after reset)
1: UART7 peripheral clock enabled

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CECEN**: HDMI-CEC clock enable
Set and reset by software.
0: HDMI-CEC peripheral clock disabled (default after reset)
1: HDMI-CEC peripheral clock enabled

Bit 27 **USART11EN**: USART11 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: USART11 peripheral clock disabled (default after reset)
1: USART11 peripheral clock enabled

Bit 26 **USART10EN**: USART10 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: USART10 peripheral clock disabled (default after reset)
1: USART10 peripheral clock enabled

Bit 25 **USART6EN**: USART6 clock enable
Set and reset by software.
0: USART6 peripheral clock disabled (default after reset)
1: USART6 peripheral clock enabled

Bit 24 **CRSEN**: CRS clock enable
Set and reset by software.
0: CRS peripheral clock disabled (default after reset)
1: CRS peripheral clock enabled

Bit 23 **I3C1EN**: I3C1 clock enable
Set and reset by software.
0: I3C1 peripheral clock disabled (default after reset)
1: I3C1 peripheral clock enabled

Bit 22 **I2C2EN**: I2C2 clock enable
Set and reset by software.
0: I2C2 peripheral clock disabled (default after reset)
1: I2C2 peripheral clock enabled

Bit 21 **I2C1EN**: I2C1 clock enable
Set and reset by software.
0: I2C1 peripheral clock disabled (default after reset)
1: I2C1 peripheral clock enabled

Bit 20 **UART5EN**: UART5 clock enable
Set and reset by software.
0: UART5 peripheral clock disabled (default after reset)
1: UART5 peripheral clock enabled

Bit 19 **UART4EN**: UART4 clock enable
Set and reset by software.
0: UART4 peripheral clock disabled (default after reset)
1: UART4 peripheral clock enabled

Bit 18 **USART3EN**: USART3 clock enable
Set and reset by software.
0: USART3 peripheral clock disabled (default after reset)
1: USART3 peripheral clock enabled

Bit 17 **USART2EN**: USART2 clock enable
Set and reset by software.
0: USART2 peripheral clock disabled (default after reset)
1: USART2 peripheral clock enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3EN**: SPI3 clock enable
Set and reset by software.
0: SPI3 peripheral clock disabled (default after reset)
1: SPI3 peripheral clock enabled

Bit 14 **SPI2EN**: SPI2 clock enable
Set and reset by software.
0: SPI2 peripheral clock disabled (default after reset)
1: SPI2 peripheral clock enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG clock enable
Set and reset by software.
0: WWDG peripheral clock disabled (default after reset)
1: WWDG peripheral clock enabled

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **TIM14EN**: TIM14 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM14 peripheral clock disabled (default after reset)
1: TIM14 peripheral clock enabled

Bit 7 **TIM13EN**: TIM13 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM13 peripheral clock disabled (default after reset)
1: TIM13 peripheral clock enabled

Bit 6 **TIM12EN**: TIM12 clock enable
Set and reset by software.
0: TIM12 peripheral clock disabled (default after reset)
1: TIM12 peripheral clock enabled

Bit 5 **TIM7EN**: TIM7 clock enable
Set and reset by software.
0: TIM7 peripheral clock disabled (default after reset)
1: TIM7 peripheral clock enabled

Bit 4 **TIM6EN**: TIM6 clock enable
Set and reset by software.
0: TIM6 peripheral clock disabled (default after reset)
1: TIM6 peripheral clock enabled

Bit 3 **TIM5EN**: TIM5 clock enable
Set and reset by software.
0: TIM5 peripheral clock disabled (default after reset)
1: TIM5 peripheral clock enabled

Bit 2 **TIM4EN**: TIM4 clock enable
Set and reset by software.
0: TIM4 peripheral clock disabled (default after reset)
1: TIM4 peripheral clock enabled

Bit 1 **TIM3EN**: TIM3 clock enable
Set and reset by software.
0: TIM3 peripheral clock disabled (default after reset)
1: TIM3 peripheral clock enabled

Bit 0 **TIM2EN**: TIM2 clock enable
Set and reset by software.
0: TIM2 peripheral clock disabled (default after reset)
1: TIM2 peripheral clock enabled

11.8.30 RCC APB1 peripheral clock register (RCC_APB1HENR)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	LPTIM2EN	Res.	DTSEN	Res.	UART12EN	UART9EN
						rw				rw		rw		rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UCPD1EN**: UCPD1 clock enable
Set and reset by software.
0: UCPD peripheral clock disabled (default after reset)
1: UCPD peripheral clock enabled

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCANEN**: FDCAN1 and FDCAN2 peripheral clock enable
Set and reset by software.
0: FDCAN1 and FDCAN2 peripheral clock disabled (default after reset)
1: FDCAN1 and FDCAN2 peripheral clock enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2EN**: LPTIM2 clock enable
Set and reset by software.
0: LPTIM2 peripheral clock disabled (default after reset)
1: LPTIM2 peripheral clock enabled

Bit 4 Reserved, must be kept at reset value.

Bit 3 **DTSEN**: DTS clock enable
Set and reset by software.
0: DTS peripheral clock disabled (default after reset)
1: DTS peripheral clock enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **UART12EN**: UART12 clock enable

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: UART12 peripheral clock disabled (default after reset)

1: UART12 peripheral clock enabled

Bit 0 **UART9EN**: UART9 clock enable

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: UART9 peripheral clock disabled (default after reset)

1: resets UART9 peripheral clock enabled

11.8.31 RCC APB2 peripheral clock register (RCC_APB2ENR)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBEN	Res.	SAI2EN	SAI1EN	SPI6EN	SPI4EN	TIM17EN	TIM16EN	TIM15EN
							rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBEN**: USB clock enable

Set and reset by software.

0: USB peripheral clock disabled (default after reset)

1: USB peripheral clock enabled

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2EN**: SAI2 clock enable

Set and cleared by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: SAI2 clock disabled

1: SAI2 clock enabled

Bit 21 **SAI1EN**: SAI1 clock enable

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: SAI1 peripheral clock disabled (default after reset)

1: SAI1 peripheral clock enabled

Bit 20 **SPI6EN**: SPI6 clock enable

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: SPI6 peripheral clock disabled (default after reset)

1: SPI6 peripheral clock enabled

Bit 19 **SPI4EN**: SPI4 clock enable
Set and reset by software.
0: SPI4 peripheral clock disabled (default after reset)
1: SPI4 peripheral clock enabled

Bit 18 **TIM17EN**: TIM17 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM17 peripheral clock disabled (default after reset)
1: TIM17 peripheral clock enabled

Bit 17 **TIM16EN**: TIM16 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM16 peripheral clock disabled (default after reset)
1: TIM16 peripheral clock enabled

Bit 16 **TIM15EN**: TIM15 clock enable
Set and reset by software.
0: TIM15 peripheral clock disabled (default after reset)
1: TIM15 peripheral clock enabled

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1EN**: USART1 clock enable
Set and reset by software.
0: USART1 peripheral clock disabled (default after reset)
1: USART1 peripheral clock enabled

Bit 13 **TIM8EN**: TIM8 clock enable
Set and reset by software.
0: TIM8 peripheral clock disabled (default after reset)
1: TIM8 peripheral clock enabled

Bit 12 **SPI1EN**: SPI1 clock enable
Set and reset by software.
0: SPI1 peripheral clock disabled (default after reset)
1: SPI1 peripheral clock enabled

Bit 11 **TIM1EN**: TIM1 clock enable
Set and reset by software.
0: TIM1 peripheral clock disabled (default after reset)
1: TIM1 peripheral clock enabled

Bits 10:0 Reserved, must be kept at reset value.

11.8.32 RCC APB3 peripheral clock register (RCC_APB3ENR)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	VREFBUFEN	Res.	Res.	Res.	Res.
										rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM6EN	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM1EN	Res.	I3C2EN	I2C4EN	I2C3EN	LPUART1EN	SPI5EN	Res.	Res.	Res.	SBSEN	Res.
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw				rw	

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

Bit 21 **RTCAPBEN**: RTC APB interface clock enable
Set and reset by software.
0: RTC APB interface clock disabled (default after reset)
1: RTC APB interface clock enabled

Bit 20 **VREFBUFEN**: VREFBUF clock enable
Set and reset by software.
0: VREFBUF peripheral clock disabled (default after reset)
1: VREFBUF peripheral clock enabled

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **LPTIM6EN**: LPTIM6 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM6 peripheral clock disabled (default after reset)
1: LPTIM6 peripheral clock enabled

Bit 14 **LPTIM5EN**: LPTIM5 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM5 peripheral clock disabled (default after reset)
1: LPTIM5 peripheral clock enabled

Bit 13 **LPTIM4EN**: LPTIM4 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM4 peripheral clock disabled (default after reset)
1: LPTIM4 peripheral clock enabled

Bit 12 **LPTIM3EN**: LPTIM3 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM3 peripheral clock disabled (default after reset)
1: LPTIM3 peripheral clock enabled

Bit 11 **LPTIM1EN**: LPTIM1 clock enable
Set and reset by software.
0: LPTIM1 peripheral clock disabled (default after reset)
1: LPTIM1 peripheral clock enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **I3C2EN**: I3C2 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: I3C2 peripheral clock disabled (default after reset)
1: I3C2 peripheral clock enabled

Bit 8 **I2C4EN**: I2C4 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: I2C4 peripheral clock disabled (default after reset)
1: I2C4 peripheral clock enabled

Bit 7 **I2C3EN**: I2C3 clock enable
Set and reset by software.
0: I2C3 peripheral clock disabled (default after reset)
1: I2C3 peripheral clock enabled

Bit 6 **LPUART1EN**: LPUART1 clock enable
Set and reset by software.
0: LPUART1 peripheral clock disabled (default after reset)
1: LPUART1 peripheral clock enabled

Bit 5 **SPI5EN**: SPI5 clock enable
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SPI5 peripheral clock disabled (default after reset)
1: SPI5 peripheral clock enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SBSEN**: SBS clock enable
Set and reset by software.
0: SBS peripheral clock disabled (default after reset)
1: SBS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.

11.8.33 RCC AHB1 sleep clock register (RCC_AHB1LPENR)

Address offset: 0x0B0

Reset value: 0xF13A D103 (for STM32H562/563/573xx)

Reset value: 0xF102 1103 (for STM32H523/533xx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM1LPEN	DCACHELPEN	ICACHELPEN	BKPRAMLPEN	Res.	Res.	Res.	TZSC1LPEN	Res.	Res.	ETHRXLPEN	ETHTXLPEN	ETHLPEN	Res.	RAMCFGLPEN	Res.
rw	rw	rw	rw				rw			rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMACLPEN	CORDICLPEN	Res.	CRCLPEN	Res.	Res.	Res.	FLITFLPEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2LPEN	GPDMA1LPEN
rw	rw		rw				rw							rw	rw

Bit 31 **SRAM1LPEN**: SRAM1 clock enable during Sleep mode
Set and reset by software
0: SRAM1 peripheral clock disabled during Sleep mode
1: SRAM1 peripheral clock enabled during Sleep mode (default after reset)

Bit 30 **DCACHELPEN**: DCACHE clock enable during Sleep mode
Set and reset by software
0: DCACHE peripheral clock disabled during Sleep mode
1: DCACHE peripheral clock enabled during Sleep mode (default after reset)

Bit 29 **ICACHELPEN**: ICACHE clock enable during Sleep mode
Set and reset by software.
0: ICACHE peripheral clock disabled during Sleep mode
1: ICACHE peripheral clock enabled during Sleep mode (default after reset)

Bit 28 **BKPRAMLPEN**: BKPRAM clock enable during Sleep mode
Set and reset by software.
0: BKPRAM peripheral clock disabled during Sleep mode
1: BKPRAM peripheral clock enabled during Sleep mode (default after reset)

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TZSC1LPEN**: TZSC1 clock enable during Sleep mode
Set and reset by software.
0: TZSC1 peripheral clock disabled during Sleep mode
1: TZSC1 peripheral clock enabled during Sleep mode (default after reset)

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **ETHRXLPEN**: ETHRX clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: ETHRX clock disabled during Sleep mode
1: ETHRX clock enabled during Sleep mode (default after reset)

Bit 20 **ETHTXLPEN**: ETHTX clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: ETHTX clock disabled during Sleep mode
1: ETHTX clock enabled during Sleep mode (default after reset)

Bit 19 **ETHLPEN**: ETH clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: ETH peripheral clock disabled during Sleep mode
1: ETH peripheral clock enabled during Sleep mode (default after reset)

Bit 18 Reserved, must be kept at reset value.

Bit 17 **RAMCFGLPEN**: RAMCFG clock enable during Sleep mode
Set and reset by software.
0: RAMCFG peripheral clock disabled during Sleep mode
1: RAMCFG peripheral clock enabled during Sleep mode (default after reset)

Bit 16 Reserved, must be kept at reset value.

Bit 15 **FMACLPEN**: FMAC clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: FMAC peripheral clock disabled during Sleep mode
1: FMAC peripheral clock enabled during Sleep mode (default after reset)

Bit 14 **CORDICLPEN**: CORDIC clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: CORDIC peripheral clock disabled during Sleep mode
1: CORDIC peripheral clock enabled during Sleep mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CRCLPEN**: CRC clock enable during Sleep mode
Set and reset by software.
0: CRC peripheral clock disabled during Sleep mode
1: CRC peripheral clock enabled during Sleep mode (default after reset)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **FLITFLPEN**: Flash interface (FLITF) clock enable during Sleep mode
Set and reset by software.
0: FLITF peripheral clock disabled during Sleep mode
1: FLITF peripheral clock enabled during Sleep mode (default after reset)

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

Bit 1 **GPDMA2LPEN**: GPDMA2 clock enable during Sleep mode
Set and reset by software.
0: GPDMA2 peripheral clock disabled during Sleep mode
1: GPDMA2 peripheral clock enabled during Sleep mode (default after reset)

Bit 0 **GPDMA1LPEN**: GPDMA1 clock enable during Sleep mode
Set and reset by software.
0: GPDMA1 peripheral clock disabled during Sleep mode
1: GPDMA1 peripheral clock enabled during Sleep mode (default after reset)

11.8.34 RCC AHB2 sleep clock register (RCC_AHB2LPENR)

Address offset: 0x0B4

Reset value: 0xC01F 1DFF (for STM32H562/563/573xx)

Reset value: 0xC01F 1CFF (for STM32H523/533xx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRAM3 LPEN	SRAM2 LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAES LPEN	PKA LPEN	RNG LPEN	HASH LPEN	AES LPEN
rw	rw										rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DCMI_ PSSI LPEN	DAC1 LPEN	ADC LPEN	Res.	GPIOI LPEN	GPIOH LPEN	GPIOG LPEN	GPIOF LPEN	GPIOE LPEN	GPIOD LPEN	GPIOC LPEN	GPIOB LPEN	GPIOA LPEN
			rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SRAM3LPEN**: SRAM3 clock enable during Sleep mode
Set and reset by software.
0: SRAM3 peripheral clock disabled during Sleep mode
1: SRAM3 peripheral clock enabled during Sleep mode (default after reset)

Bit 30 **SRAM2LPEN**: SRAM2 clock enable during Sleep mode
Set and reset by software.
0: SRAM2 peripheral clock disabled during Sleep mode
1: SRAM2 peripheral clock enabled during Sleep mode (default after reset)

Bits 29:21 Reserved, must be kept at reset value.

Bit 20 **SAESLPEN**: SAES clock enable during Sleep mode
Set and reset by software.
0: SAES peripheral clock disabled during Sleep mode
1: SAES peripheral clock enabled during Sleep mode (default after reset)

Bit 19 **PKALPEN**: PKA clock enable during Sleep mode
Set and reset by software.
0: PKA peripheral clock disabled during Sleep mode
1: PKA peripheral clock enabled during Sleep mode (default after reset)

Bit 18 **RNGLPEN**: RNG clock enable during Sleep mode
Set and reset by software.
0: RNG peripheral clock disabled during Sleep mode
1: RNG peripheral clock enabled during Sleep mode (default after reset)

Bit 17 **HASHLPEN**: HASH clock enable during Sleep mode
Set and reset by software.
0: HASH peripheral clock disabled during Sleep mode
1: HASH peripheral clock enabled during Sleep mode (default after reset)

Bit 16 **AESLPEN**: AES clock enable during Sleep mode
Set and reset by software.
0: AES peripheral clock disabled during Sleep mode
1: AES peripheral clock enabled during Sleep mode (default after reset)

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DCMI_PSSILPEN**: digital camera interface clock enable during Sleep mode (DCMI or PSSI, depending which interface is active)
Set and reset by software.
0: DCMI/PSSI peripheral clock disabled during Sleep mode
1: DCMI/PSSI peripheral clock enabled during Sleep mode (default after reset)

Bit 11 **DAC1LPEN**: DAC clock enable during Sleep mode
Set and reset by software.
0: DAC peripheral clock disabled during Sleep mode
1: DAC peripheral clock enabled during Sleep mode (default after reset)

Bit 10 **ADCLPEN**: ADC1 and 2 peripherals clock enable during Sleep mode
Set and reset by software.
0: ADC1 and 2 peripherals clock disabled during Sleep mode
1: ADC1 and 2 peripherals clock enabled during Sleep mode (default after reset)

Bit 9 Reserved, must be kept at reset value.

Bit 8 **GPIOILPEN**: GPIOI clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: GPIOI peripheral clock disabled during Sleep mode
1: GPIOI peripheral clock enabled during Sleep mode (default after reset)

Bit 7 **GPIOHLPEN**: GPIOH clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: GPIOH peripheral clock disabled during Sleep mode
1: GPIOH peripheral clock enabled during Sleep mode (default after reset)

Bit 6 **GPIOGLPEN**: GPIOG clock enable during Sleep mode
Set and reset by software.
0: GPIOG peripheral clock disabled during Sleep mode
1: GPIOG peripheral clock enabled during Sleep mode (default after reset)

Bit 5 **GPIOFLPEN**: GPIOF clock enable during Sleep mode
Set and reset by software.
0: GPIOF peripheral clock disabled during Sleep mode
1: GPIOF peripheral clock enabled during Sleep mode (default after reset)

Bit 4 **GPIOELPEN**: GPIOE clock enable during Sleep mode
Set and reset by software.
0: GPIOE peripheral clock disabled during Sleep mode
1: GPIOE peripheral clock enabled during Sleep mode (default after reset)

Bit 3 **GPIODLPEN**: GPIOD clock enable during Sleep mode
Set and reset by software.
0: GPIOD peripheral clock disabled during Sleep mode
1: GPIOD peripheral clock enabled during Sleep mode (default after reset)

Bit 2 **GPIOCLPEN**: GPIOC clock enable during Sleep mode
Set and reset by software.
0: GPIOC peripheral clock disabled during Sleep mode
1: GPIOC peripheral clock enabled during Sleep mode (default after reset)

Bit 1 **GPIOBLPEN**: GPIOB clock enable during Sleep mode
Set and reset by software.
0: GPIOB peripheral clock disabled during Sleep mode
1: GPIOB peripheral clock enabled during Sleep mode (default after reset)

Bit 0 **GPIOALPEN**: GPIOA clock enable during Sleep mode
Set and reset by software.
0: GPIOA peripheral clock disabled during Sleep mode
1: GPIOA peripheral clock enabled during Sleep mode (default after reset)

11.8.35 RCC AHB4 sleep clock register (RCC_AHB4LPENR)

Address offset: 0x0BC

Reset value: 0x0011 1880 (for STM32H562/563/573xx)

Reset value: 0x0011 0880 (for STM32H523/533xx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1 LPEN	Res.	Res.	Res.	FMC LPEN
											rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	SDMMC2 LPEN	SDMMC1 LPEN	Res.	Res.	Res.	OTFDEC1 LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw				rw							

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OCTOSPI1LPEN**: OCTOSPI1 clock enable during Sleep mode
Set and reset by software.
0: OCTOSPI1 peripheral clock disabled during Sleep mode
1: OCTOSPI1 peripheral clock enabled during Sleep mode (default after reset)

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **FMCLPEN**: FMC clock enable during Sleep mode
Set and reset by software.
0: FMC peripheral clock disabled during Sleep mode
1: FMC peripheral clock enabled during Sleep mode (default after reset)

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **SDMMC2LPEN**: SDMMC2 and SDMMC2 delay peripheral clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SDMMC2 and SDMMC2 delay peripherals clock disabled during Sleep mode
1: SDMMC2 and SDMMC2 delay peripherals clock enabled during Sleep mode (default after reset)

Bit 11 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay peripheral clock enable during Sleep mode
Set and reset by software
0: SDMMC1 and SDMMC1 delay peripherals clock disabled during Sleep mode
1: SDMMC1 and SDMMC1 delay peripherals clock enabled during Sleep mode (default after reset)

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **OTFDEC1LPEN**: OTFDEC1 clock enable during Sleep mode
Set and reset by software.
0: OTFDEC1 peripheral clock disabled during Sleep mode
1: OTFDEC1 peripheral clock enabled during Sleep mode (default after reset)

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

11.8.36 RCC APB1 sleep clock register (RCC_APB1LLPENR)

Address offset: 0x0C4

Reset value: 0xDFFE C9FF (for STM32H562/563/573xx)

Reset value: 0x13FE C87F (for STM32H523/533xx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPEN	UART7 LPEN	Res.	CEC LPEN	USART11 LPEN	USART10 LPEN	USART6 LPEN	CRS LPEN	I3C1 LPEN	I2C2 LPEN	I2C1 LPEN	UART5 LPEN	UART4 LPEN	USART3 LPEN	USART2 LPEN	Res.
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 LPEN	SPI2 LPEN	Res.	Res.	WWDG LPEN	Res.	Res.	TIM14 LPEN	TIM13 LPEN	TIM12 LPEN	TIM7 LPEN	TIM6 LPEN	TIM5 LPEN	TIM4 LPEN	TIM3 LPEN	TIM2 LPEN
rw	rw			rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8LPEN**: UART8 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: UART8 peripheral clock disabled during Sleep mode

1: UART8 peripheral clock enabled during Sleep mode (default after reset)

Bit 30 **UART7LPEN**: UART7 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: UART7 peripheral clock disabled during Sleep mode

1: UART7 peripheral clock enabled during Sleep mode (default after reset)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CECLPEN**: HDMI-CEC clock enable during Sleep mode

Set and reset by software.

0: HDMI-CEC peripheral clock disabled during Sleep mode

1: HDMI-CEC peripheral clock enabled during Sleep mode (default after reset)

Bit 27 **USART11LPEN**: USART11 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: USART11 peripheral clock disabled during Sleep mode

1: USART11 peripheral clock enabled during Sleep mode (default after reset)

Bit 26 **USART10LPEN**: USART10 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.

0: USART10 peripheral clock disabled during Sleep mode

1: USART10 peripheral clock enabled during Sleep mode (default after reset)

Bit 25 **USART6LPEN**: USART6 clock enable during Sleep mode

Set and reset by software.

0: USART6 peripheral clock disabled during Sleep mode

1: USART6 peripheral clock enabled during Sleep mode (default after reset)

Bit 24 **CRSLPEN**: CRS clock enable during Sleep mode

Set and reset by software.

0: CRS peripheral clock disabled during Sleep mode

1: CRS peripheral clock enabled during Sleep mode (default after reset)

Bit 23 **I3C1LPEN**: I3C1 clock enable during Sleep mode

Set and reset by software.

0: I3C1 peripheral clock disabled during Sleep mode

1: I3C1 peripheral clock enabled during Sleep mode (default after reset)

Bit 22 **I2C2LPEN**: I2C2 clock enable during Sleep mode
Set and reset by software.
0: I2C2 peripheral clock disabled during Sleep mode
1: I2C2 peripheral clock enabled during Sleep mode (default after reset)

Bit 21 **I2C1LPEN**: I2C1 clock enable during Sleep mode
Set and reset by software.
0: I2C1 peripheral clock disabled during Sleep mode
1: I2C1 peripheral clock enabled during Sleep mode (default after reset)

Bit 20 **UART5LPEN**: UART5 clock enable during Sleep mode
Set and reset by software.
0: UART5 peripheral clock disabled during Sleep mode
1: UART5 peripheral clock enabled during Sleep mode (default after reset)

Bit 19 **UART4LPEN**: UART4 clock enable during Sleep mode
Set and reset by software.
0: UART4 peripheral clock disabled during Sleep mode
1: UART4 peripheral clock enabled during Sleep mode (default after reset)

Bit 18 **USART3LPEN**: USART3 clock enable during Sleep mode
Set and reset by software.
0: USART3 peripheral clock disabled during Sleep mode
1: USART3 peripheral clock enabled during Sleep mode (default after reset)

Bit 17 **USART2LPEN**: USART2 clock enable during Sleep mode
Set and reset by software.
0: USART2 peripheral clock disabled during Sleep mode
1: USART2 peripheral clock enabled during Sleep mode (default after reset)

Bit 16 Reserved, must be kept at reset value.

Bit 15 **SPI3LPEN**: SPI3 clock enable during Sleep mode
Set and reset by software.
0: SPI3 peripheral clock disabled during Sleep mode
1: SPI3 peripheral clock enabled during Sleep mode (default after reset)

Bit 14 **SPI2LPEN**: SPI2 clock enable during Sleep mode
Set and reset by software.
0: SPI2 peripheral clock disabled during Sleep mode
1: SPI2 peripheral clock enabled during Sleep mode (default after reset)

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGLPEN**: WWDG clock enable during Sleep mode
Set and reset by software.
0: WWDG peripheral clock disabled during Sleep mode
1: WWDG peripheral clock enabled during Sleep mode (default after reset)

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **TIM14LPEN**: TIM14 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM14 peripheral clock disabled during Sleep mode
1: TIM14 peripheral clock enabled during Sleep mode (default after reset)

Bit 7 **TIM13LPEN**: TIM13 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM13 peripheral clock disabled during Sleep mode
1: TIM13 peripheral clock enabled during Sleep mode (default after reset)

Bit 6 **TIM12LPEN**: TIM12 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM12 peripheral clock disabled during Sleep mode
1: TIM12 peripheral clock enabled during Sleep mode (default after reset)

Bit 5 **TIM7LPEN**: TIM7 clock enable during Sleep mode
Set and reset by software.
0: TIM7 peripheral clock disabled during Sleep mode
1: TIM7 peripheral clock enabled during Sleep mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 clock enable during Sleep mode
Set and reset by software.
0: TIM6 peripheral clock disabled during Sleep mode
1: TIM6 peripheral clock enabled during Sleep mode (default after reset)

Bit 3 **TIM5LPEN**: TIM5 clock enable during Sleep mode
Set and reset by software.
0: TIM5 peripheral clock disabled during Sleep mode
1: TIM5 peripheral clock enabled during Sleep mode (default after reset)

Bit 2 **TIM4LPEN**: TIM4 clock enable during Sleep mode
Set and reset by software.
0: TIM4 peripheral clock disabled during Sleep mode
1: TIM4 peripheral clock enabled during Sleep mode (default after reset)

Bit 1 **TIM3LPEN**: TIM3 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM3 peripheral clock disabled during Sleep mode
1: TIM3 peripheral clock enabled during Sleep mode (default after reset)

Bit 0 **TIM2LPEN**: TIM2 clock enable during Sleep mode
Set and reset by software.
0: TIM2 peripheral clock disabled during Sleep mode
1: TIM2 peripheral clock enabled during Sleep mode (default after reset)

11.8.37 RCC APB1 sleep clock register (RCC_APB1HLPENR)

Address offset: 0x0C8

Reset value: 0x4080 022B (for STM32H562/563/573xx)

Reset value: 0x4080 0228 (for STM32H523/533xx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
								rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN LPEN	Res.	Res.	Res.	LPTIM2 LPEN	Res.	DTS LPEN	Res.	UART12 LPEN	UART9 LPEN
						rw				rw		rw		rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UCPD1LPEN**: UCPD1 clock enable during Sleep mode
Set and reset by software.
0: UCPD peripheral clock disabled during Sleep mode
1: UCPD peripheral clock enabled during Sleep mode (default after reset)

Bits 22:10 Reserved, must be kept at reset value.

Bit 9 **FDCANLPEN**: FDCAN1 and FDCAN2 peripheral clock enable during Sleep mode
Set and reset by software.
0: FDCAN1 and FDCAN2 peripheral clock disabled during Sleep mode
1: FDCAN1 and FDCAN2 peripheral clock enabled during Sleep mode (default after reset)

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **LPTIM2LPEN**: LPTIM2 clock enable during Sleep mode
Set and reset by software.
0: LPTIM2 peripheral clock disabled during Sleep mode
1: LPTIM2 peripheral clock enabled during Sleep mode (default after reset)

Bit 4 Reserved, must be kept at reset value.

Bit 3 **DTSLPEN**: DTS clock enable during Sleep mode
Set and reset by software.
0: DTS peripheral clock disabled during Sleep mode
1: DTS peripheral clock enabled during Sleep mode (default after reset)

Bit 2 Reserved, must be kept at reset value.

Bit 1 **UART12LPEN**: UART12 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: UART12 peripheral clock disabled during Sleep mode
1: UART12 peripheral clock enabled during Sleep mode (default after reset)

Bit 0 **UART9LPEN**: UART9 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: UART9 peripheral clock disabled during Sleep mode
1: resets UART9 peripheral clock enabled during Sleep mode (default after reset)

11.8.38 RCC APB2 sleep clock register (RCC_APB2LPENR)

Address offset: 0x0CC

Reset value: 0x017F 7800 (for STM32H562/563/573xx)

Reset value: 0x0109 7800 (for STM32H523/533xx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB LPEN	Res.	SAI2 LPEN	SAI1 LPEN	SPI6 LPEN	SPI4 LPEN	TIM17 LPEN	TIM16 LPEN	TIM15 LPEN
							rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 LPEN	TIM8 LPEN	SPI1 LPEN	TIM1 LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw											

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **USBLPEN**: USB clock enable during Sleep mode

Set and reset by software.
0: USB peripheral clock disabled during Sleep mode
1: USB peripheral clock enabled during Sleep mode (default after reset)

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SAI2LPEN**: SAI2 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SAI2 peripheral clock disabled during Sleep mode
1: SAI2 peripheral clock enabled during Sleep mode (default after reset)

Bit 21 **SAI1LPEN**: SAI1 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SAI1 peripheral clock disabled during Sleep mode
1: SAI1 peripheral clock enabled during Sleep mode (default after reset)

Bit 20 **SPI6LPEN**: SPI6 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SPI6 peripheral clock disabled during Sleep mode
1: SPI6 peripheral clock enabled during Sleep mode (default after reset)

Bit 19 **SPI4LPEN**: SPI4 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SPI4 peripheral clock disabled during Sleep mode
1: SPI4 peripheral clock enabled during Sleep mode (default after reset)

Bit 18 **TIM17LPEN**: TIM17 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM17 peripheral clock disabled during Sleep mode
1: TIM17 peripheral clock enabled during Sleep mode (default after reset)

Bit 17 **TIM16LPEN**: TIM16 clock enable during Sleep mode

Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: TIM16 peripheral clock disabled during Sleep mode
1: TIM16 peripheral clock enabled during Sleep mode (default after reset)

Bit 16 **TIM15LPEN**: TIM15 clock enable during Sleep mode
Set and reset by software.
0: TIM15 peripheral clock disabled during Sleep mode
1: TIM15 peripheral clock enabled during Sleep mode (default after reset)

Bit 15 Reserved, must be kept at reset value.

Bit 14 **USART1LPEN**: USART1 clock enable during Sleep mode
Set and reset by software.
0: USART1 peripheral clock disabled during Sleep mode
1: USART1 peripheral clock enabled during Sleep mode (default after reset)

Bit 13 **TIM8LPEN**: TIM8 clock enable during Sleep mode
Set and reset by software.
0: TIM8 peripheral clock disabled during Sleep mode
1: TIM8 peripheral clock enabled during Sleep mode (default after reset)

Bit 12 **SPI1LPEN**: SPI1 clock enable during Sleep mode
Set and reset by software.
0: SPI1 peripheral clock disabled during Sleep mode
1: SPI1 peripheral clock enabled during Sleep mode (default after reset)

Bit 11 **TIM1LPEN**: TIM1 clock enable during Sleep mode
Set and reset by software.
0: TIM1 peripheral clock disabled during Sleep mode
1: TIM1 peripheral clock enabled during Sleep mode (default after reset)

Bits 10:0 Reserved, must be kept at reset value.

11.8.39 RCC APB3 sleep clock register (RCC_APB3LPENR)

Address offset: 0x0D0

Reset value: 0x0030 F9E2 (for STM32H562/563/573xx)

Reset value: 0x0030 FAE2 (for STM32H523/533xx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPB LPEN	VREF LPEN	Res.	Res.	Res.	Res.
										rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPTIM6 LPEN	LPTIM5 LPEN	LPTIM4 LPEN	LPTIM3 LPEN	LPTIM1 LPEN	Res.	I3C2 LPEN	I2C4 LPEN	I2C3 LPEN	LPUART1 LPEN	SPI5 LPEN	Res.	Res.	Res.	SBS LPEN	Res.
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw				rw	

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

Bit 21 **RTCAPBLPEN**: RTC APB interface clock enable during Sleep mode
Set and reset by software.
0: RTC APB interface clock disabled during Sleep mode
1: RTC APB interface clock enabled during Sleep mode (default after reset)

Bit 20 **VREFLPEN**: VREFBUF clock enable during Sleep mode
Set and reset by software.
0: VREFBUF peripheral clock disabled during Sleep mode
1: VREFBUF peripheral clock enabled during Sleep mode (default after reset)

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **LPTIM6LPEN**: LPTIM6 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM6 peripheral clock disabled during Sleep mode
1: LPTIM6 peripheral clock enabled during Sleep mode (default after reset)

Bit 14 **LPTIM5LPEN**: LPTIM5 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM5 peripheral clock disabled during Sleep mode
1: LPTIM5 peripheral clock enabled during Sleep mode (default after reset)

Bit 13 **LPTIM4LPEN**: LPTIM4 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM4 peripheral clock disabled during Sleep mode
1: LPTIM4 peripheral clock enabled during Sleep mode (default after reset)

Bit 12 **LPTIM3LPEN**: LPTIM3 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: LPTIM3 peripheral clock disabled during Sleep mode
1: LPTIM3 peripheral clock enabled during Sleep mode (default after reset)

Bit 11 **LPTIM1LPEN**: LPTIM1 clock enable during Sleep mode
Set and reset by software.
0: LPTIM1 peripheral clock disabled during Sleep mode
1: LPTIM1 peripheral clock enabled during Sleep mode (default after reset)

Bit 10 Reserved, must be kept at reset value.

Bit 9 **I3C2LPEN**: I3C2 clock enable during Sleep mode
Set and reset by software.
0: I3C2 peripheral clock disabled during Sleep mode
1: I3C2 peripheral clock enabled during Sleep mode (default after reset)

Bit 8 **I2C4LPEN**: I2C4 clock enable during Sleep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: I2C4 peripheral clock disabled during Sleep mode
1: I2C4 peripheral clock enabled during Sleep mode (default after reset)

Bit 7 **I2C3LPEN**: I2C3 clock enable during Sleep mode
Set and reset by software.
0: I2C3 peripheral clock disabled during Sleep mode
1: I2C3 peripheral clock enabled during Sleep mode (default after reset)

Bit 6 **LPUART1LPEN**: LPUART1 clock enable during Sleep mode
Set and reset by software.
0: LPUART1 peripheral clock disabled during Sleep mode
1: LPUART1 peripheral clock enabled during Sleep mode (default after reset)

Bit 5 **SPI5LPEN**: SPI5 clock enable during Slsleepeep mode
Set and reset by software. For the actual availability of this bit refer to the datasheet. If not present, consider it as reserved, and keep it at reset value.
0: SPI5 peripheral clock disabled during Sleep mode
1: SPI5 peripheral clock enabled during Slesleepep mode (default after reset)

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **SBSLPEN**: SBS clock enable during Sleep mode
Set and reset by software.
0: SBS peripheral clock disabled during Sleep mode
1: SBS peripheral clock enabled during Sleep mode (default after reset)

Bit 0 Reserved, must be kept at reset value.

11.8.40 RCC kernel clock configuration register (RCC_CCIPR1)

Address offset: 0x0D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMICSEL	Res.	USART10SEL[2:0]			UART9SEL[2:0]			UART8SEL[2:0]			UART7SEL[2:0]			USART6SEL[2:1]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USART6SEL[0]	UART5SEL[2:0]			UART4SEL[2:0]			USART3SEL[2:0]			USART2SEL[2:0]			USART1SEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TIMICSEL**: TIM12, TIM15 and LPTIM2 input capture source selection
Set and reset by software.
0: No internal clock available for timers input capture (default after reset)
1: hsi_ker_ck/1024, hsi_ker_ck/8 and csi_ker_ck/128 selected for timers input capture

Bit 30 Reserved, must be kept at reset value.

Bits 29:27 **USART10SEL[2:0]**: USART10 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 26:24 **UART9SEL[2:0]**: UART9 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 23:21 **UART8SEL[2:0]**: UART8 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 20:18 **UART7SEL[2:0]**: UART7 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 17:15 **USART6SEL[2:0]**: USART6 kernel clock source selection
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 14:12 **UART5SEL[2:0]**: UART5 kernel clock source selection
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 11:9 **UART4SEL[2:0]**: UART4 kernel clock source selection
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 8:6 **USART3SEL[2:0]**: USART3 kernel clock source selection
Set and reset by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 5:3 **USART2SEL[2:0]**: USART2 kernel clock source selection
Set and reset by software.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 2:0 **USART1SEL[2:0]**: USART1 kernel clock source selection
Set and reset by software.
000: rcc_pclk2 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

11.8.41 RCC kernel clock configuration register (RCC_CCIPR2)

Address offset: 0x0DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LPTIM6SEL[2:0]			Res.	LPTIM5SEL[2:0]			Res.	LPTIM4SEL[2:0]			Res.	LPTIM3SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM2SEL[2:0]			Res.	LPTIM1SEL[2:0]			Res.	UART12SEL[2:0]			Res.	USART11SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **LPTIM6SEL[2:0]**: LPTIM6 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_r_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **LPTIM5SEL[2:0]**: LPTIM5 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_r_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **LPTIM4SEL[2:0]**: LPTIM4 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_r_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **LPTIM3SEL[2:0]**: LPTIM3 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_r_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **LPTIM2SEL[2:0]**: LPTIM2 kernel clock source selection
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_r_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_r_ck selected as kernel clock
011: lse_ker_ck selected as kernel clock
100: lsi_ker_ck selected as kernel clock
101: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **UART12SEL[2:0]**: UART12 kernel clock source selection
Set and reset by software. For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **USART11SEL[2:0]**: USART11 kernel clock source selection
Set and reset by software.For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk1 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

11.8.42 RCC kernel clock configuration register (RCC_CCIPR3)

Address offset: 0x0E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	LPUART1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	SPI6SEL[2:1]	
					rw	rw	rw							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI6SEL[0]	SPI5SEL[2:0]			SPI4SEL[2:0]			SPI3SEL[2:0]			SPI2SEL[2:0]			SPI1SEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **LPUART1SEL[2:0]**: LPUART1 kernel clock source selection
000: rcc_pclk3 s elected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: lse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:15 **SPI6SEL[2:0]**: SPI6 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk2 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: hse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 14:12 **SPI5SEL[2:0]**: SPI5 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: rcc_pclk3 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: hse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 11:9 **SPI4SEL[2:0]**: SPI4 kernel clock source selection
000: rcc_pclk2 selected as kernel clock (default after reset)
001: pll2_q_ck selected as kernel clock
010: pll3_q_ck selected as kernel clock
011: hsi_ker_ck selected as kernel clock
100: csi_ker_ck selected as kernel clock
101: hse_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 8:6 **SPI3SEL[2:0]**: SPI3 kernel clock source selection
Set and reset by software.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 5:3 **SPI2SEL[2:0]**: SPI2 kernel clock source selection
Set and reset by software.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 2:0 **SPI1SEL[2:0]**: SPI1 kernel clock source selection
Set and reset by software.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

11.8.43 RCC kernel clock configuration register (RCC_CCIPR4)

Address offset: 0x0E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	I3C2SEL[1:0]		I3C1SEL[1:0]		I2C4SEL[1:0]		I2C3SEL[1:0]		I2C2SEL[1:0]		I2C1SEL[1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2 SEL	SDMMC1 SEL	USBSEL[1:0]		SYSTICKSEL [1:0]		OCTOSPI1SEL [1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:26 **I3C2SEL[1:0]**: I3C2 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: pll3_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: no clock selected

Bits 25:24 **I3C1SEL[1:0]**: I3C1 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: pll3_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: no clock selected

Bits 23:22 **I2C4SEL[1:0]**: I2C4 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
00: rcc_pclk3 selected as kernel clock (default after reset)
01: pll3_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: csi_ker_ck selected as kernel clock

Bits 21:20 **I2C3SEL[1:0]**: I2C3 kernel clock source selection
00: rcc_pclk3 selected as kernel clock (default after reset)
01: pll3_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: csi_ker_ck selected as kernel clock

Bits 19:18 **I2C2SEL[1:0]**: I2C2 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: pll3_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: csi_ker_ck selected as kernel clock

Bits 17:16 **I2C1SEL[1:0]**: I2C1 kernel clock source selection
00: rcc_pclk1 selected as kernel clock (default after reset)
01: pll3_r_ck selected as kernel clock
10: hsi_ker_ck selected as kernel clock
11: csi_ker_ck selected as kernel clock

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **SDMMC2SEL**: SDMMC2 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
0: pll1_q_ck selected as kernel clock (default after reset)
1: pll2_r_ck selected as kernel clock

Bit 6 **SDMMC1SEL**: SDMMC1 kernel clock source selection
0: pll1_q_ck selected as kernel clock (default after reset)
1: pll2_r_ck selected as kernel clock

Bits 5:4 **USBSEL[1:0]**: USB kernel clock source selection
00: no clock is selected as kernel clock (default after reset)
01: pll1_q_ck selected as kernel clock
10: pll3_q_ck selected as kernel clock
11: hsi48_ker_ck selected as kernel clock

Bits 3:2 **SYSTICKSEL[1:0]**: SYSTICK clock source selection
00: rcc_hclk/8 selected as clock source (default after reset)
01: lsi_ker_ck[1] selected as clock source
10: lse_ck[1] selected as clock source
11: reserved, the kernel clock is disabled
*Note: rcc_hclk frequency must be four times higher than lsi_ker_ck/lse_ck, that is, period (LSI/LSE) ≥ 4 * period (HCLK).*

Bits 1:0 **OCTOSPI1SEL[1:0]**: OCTOSPI1 kernel clock source selection
Set and reset by software.
00: rcc_hclk4 selected as kernel clock (default after reset)
01: pll1_q_ck selected as kernel clock
10: pll2_r_ck selected as kernel clock
11: per_ck selected as kernel clock

11.8.44 RCC kernel clock configuration register (RCC_CCIPR5)

Address offset: 0x0E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKPERSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2SEL[2:0]			SAI1SEL[2:0]		
rw	rw									rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FDCAN SEL[1:0]		CECSEL[1:0]		RNGSEL[1:0]		DACSEL	ADCDACSEL[2:0]		
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **CKPERSEL[1:0]**: per_ck clock source selection
00: hsi_ker_ck selected as kernel clock (default after reset)
01: csi_ker_ck selected as kernel clock
10: hse_ck selected as kernel clock
11: reserved, the per_ck clock is disabled

Bits 29:22 Reserved, must be kept at reset value.

Bits 21:19 **SAI2SEL[2:0]**: SAI2 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 18:16 **SAI1SEL[2:0]**: SAI1 kernel clock source selection
For the actual availability of these bits refer to the datasheet. If not present, consider them as reserved, and keep them at reset value.
000: pll1_q_ck selected as kernel clock (default after reset)
001: pll2_p_ck selected as kernel clock
010: pll3_p_ck selected as kernel clock
011: AUDIOCLK selected as kernel clock
100: per_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FDCANSEL[1:0]**: FDCAN1 and FDCAN2 kernel clock source selection
00: hse_ck selected as kernel clock (default after reset)
01: pll1_q_ck selected as kernel clock
10: pll2_q_ck selected as kernel clock
11: reserved, the kernel clock is disabled

Bits 7:6 **CECSEL[1:0]**: HSMI-CEC kernel clock source selection
00: lse_ck selected as kernel clock (default after reset)
01: lsi_ker_ck selected as kernel clock
10: csi_ker_ck/122 selected as kernel clock
11: reserved, the kernel clock is disabled

Bits 5:4 **RNGSEL[1:0]**: RNG kernel clock source selection
00: hsi48_ker_ck selected as kernel clock (default after reset)
01: pll1_q_ck selected as kernel clock
10: lse_ck selected as kernel clock
11: lsi_ker_ck selected as kernel clock

Bit 3 **DACSEL**: DAC sample and hold clock source selection
This bit is used to select the DAC sample and hold clock source (dac_hold_ck)
0: LSE selected
1: LSI selected

Bits 2:0 **ADCDACSEL[2:0]**: ADC and DAC kernel clock source selection
000: rcc_hclk selected as kernel clock (default after reset)
001: sys_ck selected as kernel clock
010: pll2_r_ck selected as kernel clock
011: hse_ck selected as kernel clock
100: hsi_ker_ck selected as kernel clock
101: csi_ker_ck selected as kernel clock
Others: reserved, the kernel clock is disabled

11.8.45 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x0F0

Reset value: 0x0000 0000

Reset by Backup domain reset.

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access.

Wait states are inserted in case of successive accesses to this register.

Note: After a system reset, this register is write-protected (except bits 27:24 and bit 16). To modify the backup domain bits, the DBP bit in the PWR Backup domain control register (PWR_DBPCR) must be set to 1. All bits (except bits 27:24 and bit 16) are reset only after a Backup domain reset (see Section 11.3.3). Other resets do not have effect on these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSI RDY	LSI ON	LSCO SEL	LSCO EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSW RST
				rw	rw	rw	rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC EN	Res.	Res.	Res.	Res.	Res.	RTCSEL[1:0]		LSE EXT	LSE CSSD	LSE CSSON	LSEDRV[1:0]		LSE BYP	LSE RDY	LSE ON
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **LSIRDY**: LSI oscillator ready
Set and cleared by hardware to indicate when the LSI oscillator is stable. After the LSION bit is cleared, LSIRDY goes low after three internal low-speed oscillator clock cycles.
This bit is set when the LSI is used by IWDG or RTC, even if LSION = 0.
0: LSI oscillator not ready
1: LSI oscillator ready

Bit 26 **LSION**: LSI oscillator enable
Set and cleared by software.
0: LSI oscillator off
1: LSI oscillator on

Bit 25 **LSCOSEL**: Low-speed clock output selection
Set and cleared by software.
0: LSI clock selected
1: LSE clock selected

Bit 24 **LSCOEN**: Low-speed clock output (LSCO) enable
Set and cleared by software.
0: LSCO output disabled
1: LSCO output enabled

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

Bit 16 **VSWRST**: VSwitch domain software reset
Set and reset by software.
0: reset not activated (default after Backup domain reset)
1: resets the entire VSW domain

Bit 15 **RTCEN**: RTC clock enable
Set and reset by software.
0: rtc_ck disabled (default after Backup domain reset)
1: rtc_ck enabled

Bits 14:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**:RTC clock source selection
Set by software to select the clock source for the RTC.
These bits can be written only one time (except in case of failure detection on LSE).
These bits must be written before LSECSSON is enabled.
The VSWRST bit can be used to reset them, then it can be written one time again.
If HSE is selected as RTC clock, this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).
00: no clock (default after Backup domain reset)
01: LSE selected as RTC clock
10: LSI selected as RTC clock
11: HSE divided by RTCPRE value selected as RTC clock

Bit 7 **LSEEXT**: low-speed external clock type in bypass mode
Set and reset by software to select the external clock type (analog or digital).
The external clock must be enabled with the LSEON bit, to be used by the device.
The LSEEXT bit can be written only if the LSE oscillator is disabled.
0: LSE in analog mode (default after Backup domain reset)
1: LSE in digital mode (do not use if RTC is active).

Bit 6 **LSECSSD**: LSE clock security system failure detection
Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator.
0: no failure detected on 32 kHz oscillator (default after Backup domain reset)
1: failure detected on 32 kHz oscillator

Bit 5 **LSECSSON**: LSE clock security system enable
Set by software to enable the clock security system on 32 kHz oscillator.
LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware) and after RTCSEL is selected.
Once enabled, this bit cannot be disabled, except after a LSE failure detection (LSECSSD = 1). In that case the software must disable LSECSSON.
0: CSS on 32 kHz oscillator OFF (default after Backup domain reset)
1: CSS on 32 kHz oscillator ON

Bits 4:3 **LSEDRV[1:0]**: LSE oscillator driving capability
Set by software to select the driving capability of the LSE oscillator.
These bit can be written only if LSE oscillator is disabled (LSEON = 0 and LSERDY = 0).
00: lowest drive (default after Backup domain reset)
01: medium-low drive
10: medium-high drive
11: highest drive

Bit 2 **LSEBYP**: LSE oscillator bypass
Set and reset by software to bypass oscillator in debug mode. This bit must not be written when the LSE is enabled (by LSEON) or ready (LSERDY = 1)
0: LSE oscillator not bypassed (default after Backup domain reset)
1: LSE oscillator bypassed

Bit 1 **LSERDY**: LSE oscillator ready
Set and reset by hardware to indicate when the LSE is stable.
This bit needs 6 cycles of lse_ck clock to fall down after LSEON has been set to 0.
0: LSE oscillator not ready (default after Backup domain reset)
1: LSE oscillator ready

Bit 0 **LSEON**: LSE oscillator enabled
Set and reset by software.
0: LSE oscillator OFF (default after Backup domain reset)
1: LSE oscillator ON

11.8.46 RCC reset status register (RCC_RSR)

Address offset: 0x0F4

Reset value: 0x0C00 0000

Reset by power-on reset only.

Access: 0 ≤ wait state ≤ 3, word, half-word and byte access.

Wait states are inserted in case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPWR RSTF	WWDG RSTF	IWDG RSTF	SFT RSTF	BOR RSTF	PIN RSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r			rw							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **LPWRRSTF**: Low-power reset flag
Set by hardware when a reset occurs due to Stop or Standby mode entry, whereas the corresponding nRST_STOP, nRST_STBY option bit is cleared.
Cleared by writing to the RMVF bit.
0: No illegal low-power mode reset occurred
1: Illegal low-power mode reset occurred

Bit 30 **WWDGRSTF**: window watchdog reset flag
Reset by software by writing the RMVF bit.
Set by hardware when a window watchdog reset occurs.
0: no window watchdog reset occurred from WWDG (default after power-on reset)
1: window watchdog reset occurred from WWDG

Bit 29 **IWDGRSTF**: independent watchdog reset flag
Reset by software by writing the RMVF bit.
Set by hardware when an independent watchdog reset occurs.
0: no independent watchdog reset occurred (default after power-on reset)
1: independent watchdog reset occurred

Bit 28 **SFTRSTF**: system reset from CPU reset flag
Reset by software by writing the RMVF bit.
Set by hardware when the system reset is due to CPU.The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the core M33.
0: no CPU software reset occurred (default after power-on reset)
1: a system reset has been generated by the CPU

Bit 27 **BORRSTF**: BOR reset flag
Reset by software by writing the RMVF bit.
Set by hardware when a BOR reset occurs (pwr_bor_rst).
0: no BOR reset occurred
1: BOR reset occurred (default after power-on reset)

Bit 26 **PINRSTF**: pin reset flag (NRST)
Reset by software by writing the RMVF bit.
Set by hardware when a reset from pin occurs.
0: no reset from pin occurred
1: reset from pin occurred (default after power-on reset)

Bits 25:24 Reserved, must be kept at reset value.

Bit 23 **RMVF**: remove reset flag
Set and reset by software to reset the value of the reset flags.
0: reset of the reset flags not activated (default after power-on reset)
1: resets the value of the reset flags

Bits 22:0 Reserved, must be kept at reset value.

11.8.47 RCC secure configuration register (RCC_SECCFGR)

Address offset: 0x110

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

When the system is secure (TZEN = 0xB4), this register can be written only by a secure privileged access if SPRIV = 1, and by a secure privileged or unprivileged access if SPRIV = 0. A non-secure write access generates an illegal access event and data is not written. This register can be read by secure or non-secure, privilege or unprivileged access. When the system is not secure (TZEN = 0xC3), this register is read as 0 and the register write is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CKPERSELSEC	RMVFSEC	HSI48SEC	Res.	PLL3SEC	PLL2SEC	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	CSISEC	HSESEC	HSISEC
		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **CKPERSELSEC**: per_ck selection security
Set and reset by software.
0: non secure
1: secure

Bit 12 **RMVFSEC**: Remove reset flag security
Set and reset by software.
0: non secure
1: secure

Bit 11 **HSI48SEC**: HSI48 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 10 Reserved, must be kept at reset value.

Bit 9 **PLL3SEC**: PLL3 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 8 **PLL2SEC**: PLL2 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 7 **PLL1SEC**: PLL1 clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 6 **PRESCSEC**: AHBx/APBx prescaler configuration bits security
Set and reset by software.
0: non secure
1: secure

Bit 5 **SYSCLKSEC**: SYSCLK clock selection, STOPWUCK bit, clock output on MCO configuration security
Set and reset by software.
0: non secure
1: secure

Bit 4 **LSESEC**: LSE clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 3 **LSISEC**: LSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 2 **CSISEC**: CSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

Bit 1 **HSESEC**: HSE clock configuration bits, status bits and HSE_CSS security
Set and reset by software.
0: non secure
1: secure

Bit 0 **HSISEC**: HSI clock configuration and status bits security
Set and reset by software.
0: non secure
1: secure

11.8.48 RCC privilege configuration register (RCC_PRIVCFGR)

Address offset: 0x114

Reset value: 0x0000 0000

Access: no wait state; word, half-word and byte access

This register can be written only by a privileged access. It can be read by privileged or unprivileged access.

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

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

Bit 1 **NSPRIV**: RCC non-secure functions privilege configuration
Set and reset by software. This bit can be written only by privileged access, secure or non-secure.
0: Read and write to RCC non-secure functions can be done by privileged or unprivileged access.
1: Read and write to RCC non-secure functions can be done by privileged access only

Bit 0 **SPRIV**: RCC secure functions privilege configuration
Set and reset by software. This bit can be written only by a secure privileged access.
0: Read and write to RCC secure functions can be done by privileged or unprivileged access.
1: Read and write to RCC secure functions can be done by privileged access only

11.9 RCC register map

Table 118. RCC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Res.	Res.	PLL3RDY	PLL3ON	PLL2RDY	PLL2ON	PLL1RDY	PLL1ON	Res.	Res.	Res.	HSEEXT	HSECSSON	HSEBYP	HSERDY	HSEON	Res.	Res.	HSI48RDY	HSI48ON	Res.	CSIKERON	CSIRDY	CSION	Res.	Res.	HSIDIVF	HSIDIV [1:0]		HSIKERON	HSIRDY	HSION
	Reset value			0	0	0	0	0	0				0	0	0	0	0			0	0		0	0	0			1	0	1	0	1	1
0x010	RCC_HSICFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSITRIM[6:0]							Res.	Res.	Res.	Res.	HSICAL[11:0]											
	Reset value										1	0	0	0	0	0	0					X	X	X	X	X	X	X	X	X	X	X	X
0x014	RCC_CRRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSI48CAL[9:0]									
	Reset value																							X	X	X	X	X	X	X	X	X	X
0x018	RCC_CSICFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSITRIM[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSICAL[7:0]							
	Reset value											1	0	0	0	0	0									X	X	X	X	X	X	X	X
0x01C	RCC_CFGR1	MCO2SEL[2:0]		MCO2PRE[3:0]				MCO1SEL[2:0]			MCO1PRE[3:0]					Res.	Res.	TIMPRE	Res.	RTCPRE[5:0]						STOPKERWUCK	STOPWUCK	Res.	SWS [1:0]		Res.	SW [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0		0	0	0	0	0	0	0	0		0	0		0	0
0x020	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APB3DIS	APB2DIS	APB1DIS	AHB4DIS	Res.	AHB2DIS	AHB1DIS	Res.	Res.	PPRE3 [2:0]		Res.	PPRE2 [2:0]			Res.	PPRE1 [2:0]			HPRE [3:0]			
	Reset value										0	0	0	0		0	0			0	0	0		0	0	0	0	0	0	0	0	0	0
0x028	RCC_PLL1CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1REN	PLL1QEN	PLL1PEN	Res.	Res.	PLL1M[5:0]						Res.	Res.	PLL1VCOSEL	PLL1FRACEN	PLL1RGE [1:0]		PLL1SRC [1:0]	
	Reset value														0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x02C	RCC_PLL2CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2REN	PLL2QEN	PLL2PEN	Res.	Res.	PLL2M[5:0]						Res.	Res.	PLL2VCOSEL	PLL2FRACEN	PLL2RGE [1:0]		PLL2SRC [1:0]	
	Reset value														0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x030	RCC_PLL3CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3REN	PLL3QEN	PLL3PEN	Res.	Res.	PLL3M[5:0]						Res.	Res.	PLL3VCOSEL	PLL3FRACEN	PLL3RGE [1:0]		PLL3SRC [1:0]	
	Reset value														0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x034	RCC_PLL1DIVR	Res.	PLL1R[6:0]							Res.	PLL1Q[6:0]							PLL1P[6:0]						PLL1N[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x038	RCC_PLL1FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			

Table 118. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03C	RCC_PLL2DIVR	Res.	PLL2R[6:0]							Res.	PLL2Q[6:0]							PLL2P[6:0]							PLL2N[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x040	RCC_PLL2FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x044	RCC_PLL3DIVR	Res.	PLL3R[6:0]							Res.	PLL3Q[6:0]							PLL3P[6:0]							PLL3N[8:0]								
	Reset value		0	0	0	0	0	0	1		0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
0x048	RCC_PLL3FRACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3FRACN[12:0]													Res.	Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x050	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	HSI48RDYIE	HSERDYIE	HSIRDYIE	CSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value																								0	0	0	0	0	0	0	0	0
0x054	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSF	Res.	PLL3RDYF	PLL2RDYF	PLL1RDYF	HSI48RDYF	HSERDYF	HSIRDYF	CSIRDYF	LSERDYF	LSIRDYF
	Reset value																						0		0	0	0	0	0	0	0	0	0
0x058	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSC	Res.	PLL3RDYC	PLL2RDYC	PLL1RDYC	HSI48RDYC	HSERDYC	HSIRDYC	CSIRDYC	LSERDYC	LSIRDYC
	Reset value																						0		0	0	0	0	0	0	0	0	0
0x060	RCC_AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETHRST[1]	Res.	RAMCFGRST	Res.	FMACRST[1]	CORDICRST[1]	Res.	CRCRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2RST	GPDMA1RST
	Reset value													0		0		0	0		0											0	0
0x064	RCC_AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAESRST	PKARST	RNGRST	HASHRST	AESRST	Res.	Res.	Res.	DCMI_PSSIRST	DAC1RST	ADCRST	Res.	GPIOIRST[1]	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
	Reset value												0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0	0
0x06C	RCC_AHB4RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1RST	Res.	Res.	Res.	FMCRST	Res.	Res.	Res.	SDMMC2RST[1]	SDMMC1RST	Res.	Res.	Res.	OTFDEC1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0				0				0	0				0							
0x074	RCC_APB1LRSTR	UART8RST[1]	UART7RST[1]	Res.	CECRST	USART11[1]	USART10RST[1]	USART6RST	CRSRST	I3C1RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	Res.	SPI3RST	SPI2RST	Res.	Res.	Res.	Res.	Res.	TIM14RST[1]	TIM13RST[1]	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0		0	0						0	0	0	0	0	0	0	0	0

Table 118. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x078	RCC_APB1HRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRST	Res.	Res.	Res.	LPTIM2RST	Res.	DTSRST	Res.	UART12RST(1)	UART9RST(1)
	Reset value									0														0				0		0		0	0
0x07C	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBRST	Res.	SAI2RST(1)	SAI1RST(1)	SPI6RST(1)	SPI4RST	TIM17RST(1)	TIM16RST(1)	TIM15RST	Res.	USART1RST	TIM8RST	SPI1RST	TIM1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0	0	0	0	0	0	0		0	0	0	0											
0x080	RCC_APB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VREFRST	Res.	Res.	Res.	Res.	LPTIM6RST(1)	LPTIM5RST(1)	LPTIM4RST(1)	LPTIM3RST(1)	LPTIM1RST	Res.	I3C2RST(1)	I2C4RST(1)	I2C3RST	LPUART1RST	SPI5RST(1)	Res.	Res.	Res.	Res.	Res.
	Reset value												0					0	0	0	0	0		0	0	0	0	0					
0x088	RCC_AHB1ENR	SRAM1EN	DCACHEEN	Res.	BKPRAMEN	Res.	Res.	Res.	TZSC1EN	Res.	Res.	ETHRXEN(1)	ETHTXEN(1)	ETHEN(1)	Res.	RAMCFGEN	Res.	FMACEN(1)	CORDICEN(1)	Res.	CRCEN	Res.	Res.	Res.	FLITFEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2EN	GPDMA1EN
	Reset value	1	1		1				0			0	0	0		0		0	0		0				1							0	0
0x08C	RCC_AHB2ENR	SRAM3EN	SRAM2EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAESEN	PKAEN	RNGEN	HASHEN	AESEN	Res.	Res.	Res.	DCMI_PSSIEN	DAC1EN	ADCEN	Res.	GPIOIEN(1)	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value	1	1										0	0	0	0	0				0	0	0		0	0	0	0	0	0	0	0	0
0x094	RCC_AHB4ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1EN	Res.	Res.	Res.	FMCEN	Res.	Res.	Res.	SDMMC2EN(1)	SDMMC1EN	Res.	Res.	Res.	OTFDEC1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0				0				0	0				0								
0x09C	RCC_APB1LENR	UART8EN(1)	UART7EN(1)	Res.	CECEN	USART11(1)	USART10EN(1)	USART6EN	CRSEN	I3C1EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	Res.	SPI3EN	SPI2EN	Res.	Res.	WWDGEN	Res.	Res.	TIM14EN(1)	TIM13EN(1)	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0		0	0			0			0	0	0	0	0	0	0	0	0
0x0A0	RCC_APB1HENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	Res.	LPTIM2EN	Res.	DTSEN	Res.	UART12EN(1)	UART9EN(1)
	Reset value									0														0				0		0		0	0
0x0A4	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBEN	Res.	SAI2EN(1)	SAI1EN(1)	SPI6EN(1)	SPI4EN	TIM17EN(1)	TIM16EN(1)	TIM15EN	Res.	USART1EN	TIM8EN	SPI1EN	TIM1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0	0	0	0	0	0	0		0	0	0	0											

Table 118. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0A8	RCC_APB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	VREFEN	Res.	Res.	Res.	Res.	LPTIM6EN(1)	LPTIM5EN(1)	LPTIM4EN(1)	LPTIM3EN(1)	LPTIM1EN	Res.	I3C2EN(1)	I2C4EN(1)	I2C3EN	LPUART1EN	SPI5EN(1)	Res.	Res.	Res.	SBSEN	Res.
	Reset value											0	0					0	0	0	0	0		0	0	0	0	0				0	
0x0B0	RCC_AHB1LPENR	SRAM1LPEN	DCACHELPEN	ICACHELPEN	BKPRAMLPEN	Res.	Res.	Res.	TZSC1LPEN	Res.	Res.	ETHRXLPEN(2)	ETHTXLPEN(2)	ETHLPEN(2)	Res.	RAMCFGLPEN	Res.	FMACLPEN(2)	CORDICLPEN(2)	Res.	CRCLPEN	Res.	Res.	Res.	FLITFLPEN	Res.	Res.	Res.	Res.	Res.	Res.	GPDMA2LPEN	GPDMA1LPEN
	Reset value	1	1	1	1				1			1	1	1		1		1	1		1				1							1	1
0x0B4	RCC_AHB2LPENR	SRAM3LPEN	SRAM2LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAESLPEN	PKALPEN	RNGLPEN	HASHLPEN	AESLPEN	Res.	Res.	Res.	DCMI_PSSILPEN	DAC1LPEN	ADCLPEN	Res.	GPIOILPEN	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value	1	1										1	1	1	1	1				1	1	1		1	1	1	1	1	1	1	1	1
0x0BC	RCC_AHB4LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCTOSPI1LPEN	Res.	Res.	Res.	FMCLPEN	Res.	Res.	Res.	SDMMC2LPEN(2)	SDMMC1LPEN	Res.	Res.	Res.	OTFDEC1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												1				1				1	1				1							
0x0C4	RCC_APB1LLPENR	UART8LPEN(2)	UART7LPEN(2)	Res.	CECLPEN	USART11LPEN(2)	USART10LPEN(2)	USART6LPEN	CRSLPEN	I3C1LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	Res.	SPI3LPEN	SPI2LPEN	Res.	Res.	WWDGLPEN	Res.	Res.	TIM14LPEN(2)	TIM13LPEN(2)	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	1	1		1	1	1	1	1	1	1	1	1	1	1	1		1	1			1			1	1	1	1	1	1	1	1	1
0x0C8	RCC_APB1HLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	Res.	LPTIM2LPEN	Res.	DTSLPEN	Res.	UART12LPEN(2)	UART9LPEN(2)
	Reset value									1														1				1		1		1	1
0x0CC	RCC_APB2LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USBLPEN	Res.	SAI2LPEN(2)	SAI1LPEN(2)	SPI6LPEN(2)	SPI4LPEN	TIM17LPEN(2)	TIM16LPEN(2)	TIM15LPEN	Res.	USART1LPEN	TIM8LPEN	SPI1LPEN	TIM1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								1		1	1	1	1	1	1	1		1	1	1	1												
0x0D0	RCC_APB3LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN	VREFLPEN	Res.	Res.	Res.	Res.	LPTIM6LPEN(2)	LPTIM5LPEN(2)	LPTIM4LPEN(2)	LPTIM3LPEN(2)	LPTIM1LPEN	Res.	I3C2LPEN(3)	I2C4LPEN(2)	I2C3LPEN	LPUART1LPEN	SPI5LPEN(2)	Res.	Res.	Res.	SBSLPEN	Res.
	Reset value											1	1					1	1	1	1	1		0	1	1	1	1				1	

Table 118. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D8	RCC_CCIPR1	TIMICSEL	Res.	USART10SEL [2:0](4)			UART9SEL [2:0](4)			UART8SEL [2:0](4)			UART7SEL [2:0](4)			USART6SEL [2:0]			UART5SEL [2:0]			UART4SEL [2:0]			USART3SEL [2:0]			USART2SEL [2:0]			USART1SEL [2:0]		
	Reset value	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DC	RCC_CCIPR2	Res.	LPTIM6SEL [2:0](4)			Res.	LPTIM5SEL [2:0](4)			Res.	LPTIM4SEL [2:0](4)			Res.	LPTIM3SEL [2:0](4)			Res.	LPTIM2SEL [2:0]			Res.	LPTIM1SEL [2:0]			Res.	UART12SEL [2:0](4)			Res.	USART11SEL [2:0](4)		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x0E0	RCC_CCIPR3	Res.	Res.	Res.	Res.	Res.	LPUART1SEL [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	SPI6SEL [2:0](4)			SPI5SEL [2:0](4)			SPI4SEL [2:0]			SPI3SEL [2:0]			SPI2SEL [2:0]			SPI1SEL [2:0]		
	Reset value						0	0	0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E4	RCC_CCIPR4	Res.	Res.	Res.	Res.	Res.	Res.	I3C1SEL [1:0]		I2C4SEL [1:0](4)		I2C3SEL [1:0]		I2C2SEL [1:0]		I2C1SEL [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2SEL(1)	SDMMC1SEL	USBSEL [1:0]		SYSTICKSEL [1:0]		OCTOSPI [1:0]	
	Reset value							0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x0E8	RCC_CCIPR5	CKPERSEL [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2SEL [2:0](4)			SAI1SEL [2:0](4)			Res.	Res.	Res.	Res.	Res.	Res.	FDCANSEL [1:0]		CECSEL [1:0]		RNGSEL [1:0]		DACSEL		ADCDACSEL [2:0]	
	Reset value	0	0									0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0F0	RCC_BDCR	Res.	Res.	Res.	Res.	LSIRDY	LSION	LSCOSEL	LSCOEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSWRST	RTCEN	Res.	Res.	Res.	Res.	Res.	Res.	RTCSEL [1:0]		LSEEXT	LSECSSD	LSECSSON	LSEDRV [1:0]		LSEBYP	LSERDY	LSEON
	Reset value					0	0	0	0								0	0						0	0	0	0	0	0	0	0	0	0
0x0F4	RCC_RSR	LPWRRSTF	WWDGRSTF	IWDGRSTF	SFTRSTF	BORRSTF	PINRSTF	Res.	Res.	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	1	1			0																							
0x110	RCC_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKPERSELSEC	RMVFSEC	HSI48SEC	Res.	PLL3SEC	PLL2SEC	PLL1SEC	PRESCSEC	SYSCLKSEC	LSESEC	LSISEC	CSISEC	HSESEC	HSISEC
	Reset value																			0	0	0		0	0	0	0	0	0	0	0	0	0
0x114	RCC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NSPRIV	SPRI
	Reset value																															0	0

1. For the actual availability of this bit check the datasheet. If not present, consider it as reserved, and keep it at reset value.
2. Reset value = 0 for STM32H523/533xx devices.
3. Reset value = 1 for STM32H523/533xx devices.

4. For the actual availability of these bits check the datasheet. If not present, consider them as reserved, and keep them at reset value.

Refer to *Section 2.3* for the register boundary addresses.

12 Clock recovery system (CRS)

12.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from alternative synchronization or generated by the user software.

12.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity (see *Section 12.4.2: CRS internal signals*)
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

12.3 CRS implementation

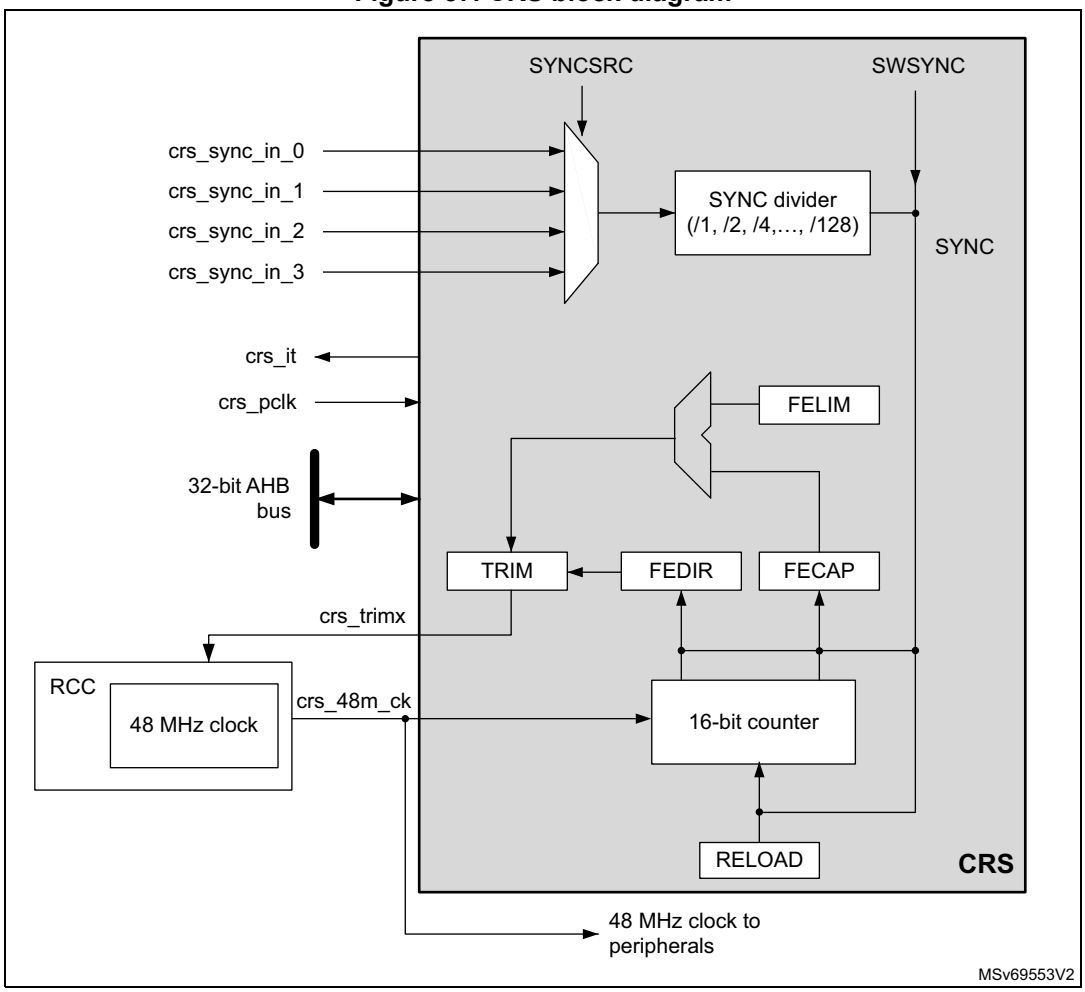
Table 119. CRS features

Feature	CRS
TRIM width	6 bits

12.4 CRS functional description

12.4.1 CRS block diagram

Figure 57. CRS block diagram



12.4.2 CRS internal signals

Below the list of CRS internal signals.

Table 120. CRS internal input/output signals

Internal signal name	Signal type	Description
crs_it	Digital output	CRS interrupt
crs_pclk	Digital input	AHB bus clock
crs_48m_ck	Digital input	HSI48 48 MHz clock
crs_trim[0:5]	Digital output	HSI48 oscillator smooth trimming value

Table 120. CRS internal input/output signals (continued)

Internal signal name	Signal type	Description
crs_sync_in_0	Digital input	SYNC signal source
crs_sync_in_1		
crs_sync_in_2		
crs_sync_in_3		

Table 121. CRS interconnection

Internal signal name	Description
crs_sync_in_0	GPIO selected as SYNC signal source
crs_sync_in_1	LSE selected as SYNC signal source
crs_sync_in_2	USB SOF selected as SYNC signal source (default)
crs_sync_in_3	Reserved

12.4.3 Synchronization input

The CRS synchronization source (crs_sync_in_x) can be selected through SYNCSRC[1:0] bitfield of CRS_CFGR register (refer to *Section 12.4.2: CRS internal signals* for the possible sources). For a better robustness of the crs_sync_in_x input, a simple digital filter (2 out of 3 majority votes, sampled by the 48 MHz clock) is implemented to filter out glitches. In addition, the source signal has a configurable polarity (selected through SYNCPOL bit of CRS_CFGR). The signal can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to CRS_CFGR register.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

For more information on the CRS synchronization source configuration, refer to *CRS configuration register (CRS_CFGR)*.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

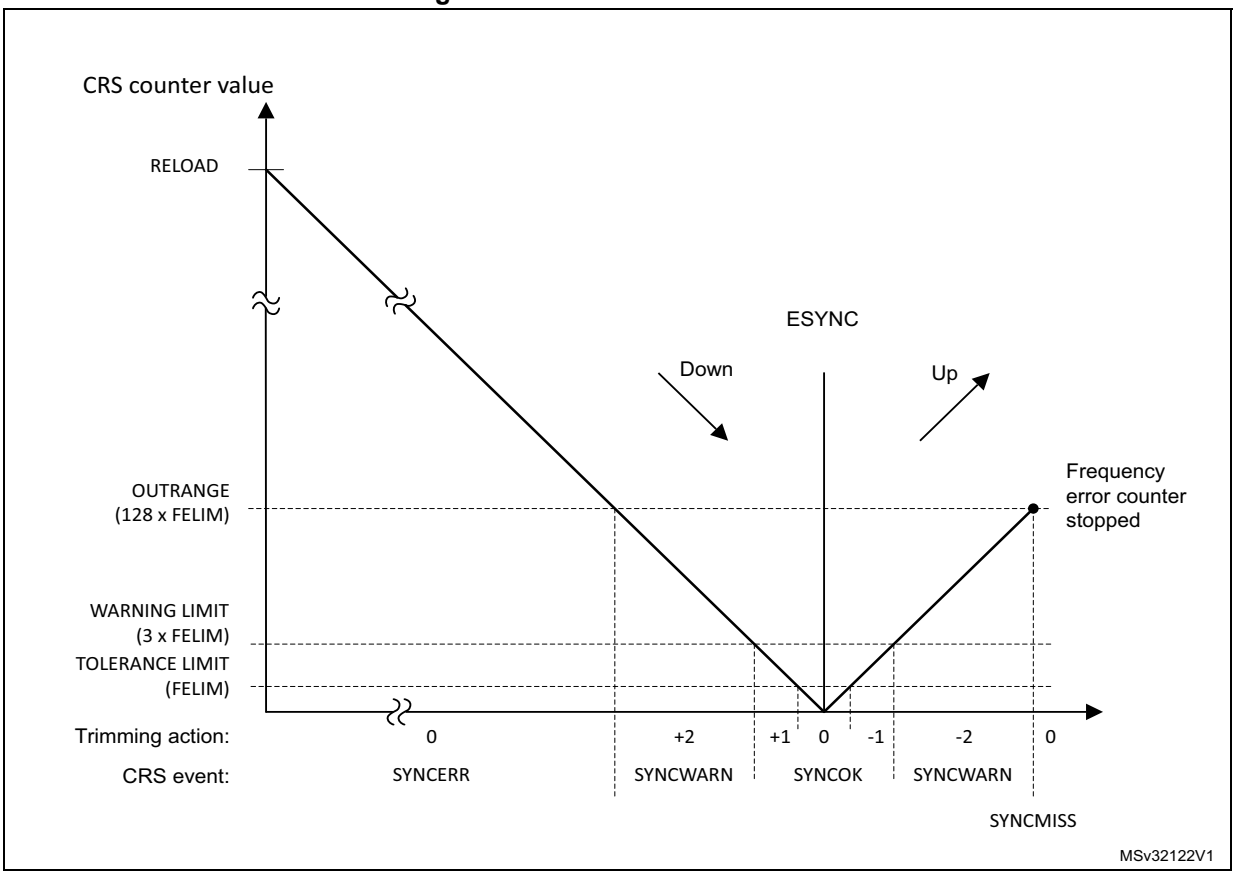
12.4.4 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual

frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).

Figure 58. CRS counter behavior



12.4.5 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:

- TOLERANCE LIMIT, given directly in the FELIM field of the CRS_CFGR register
- WARNING LIMIT, defined as 3 × FELIM value
- OUTRANGE (error limit), defined as 128 × FELIM value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:

- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated

  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

Note: If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

12.4.6 CRS initialization and configuration

RELOAD value

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

$$RELOAD = (f_{TARGET} / f_{SYNC}) - 1$$

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

FELIM value

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

$$FELIM = (f_{TARGET} / f_{SYNC}) * STEP[\%] / 100\% / 2$$

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to $(f_{TARGET} / f_{SYNC}) = 48000$, and to a typical trimming step size of 0.14%.

Note: The trimming step size depends upon the product, check the datasheet for accurate setting.

Caution: There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).

12.5 CRS in low-power modes

Table 122. Effect of low-power modes on CRS

Mode	Description
Sleep	No effect. CRS interrupts cause the device to exit the Sleep mode.
Stop	CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.
Standby	The peripheral is powered down and must be reinitialized after exiting Standby mode.
Shutdown	The peripheral is powered down and must be reinitialized after exiting Shutdown mode.

12.6 CRS interrupts

Table 123. Interrupt control bits

Interrupt event	Event flag	Enable control bit	Clear flag bit
Expected synchronization	ESYNCF	ESYNCIE	ESYNCC
Synchronization OK	SYNCOKF	SYNCOKIE	SYNCOKC
Synchronization warning	SYNCWARNF	SYNCWARNIE	SYNCWARNC
Synchronization or trimming error (TRIMOVF, SYNCMISS, SYNCERR)	ERRF	ERRIE	ERRC

12.7 CRS registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

12.7.1 CRS control register (CRS_CR)

Address offset: 0x00

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	TRIM[5:0]						SW SYNC	AUTO TRIMEN	CEN	Res.	ESYNC IE	ERR IE	SYNC WARNIE	SYNC OKIE
		rw	rw	rw	rw	rw	rw	rt_w1	rw	rw		rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.

The default value is 32, corresponding to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action

1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to *Section 12.4.5* for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.

1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled

1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCIE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled

1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled

1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCWARNIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled

1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCOKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled

1: SYNC event OK (SYNCOKF) interrupt enabled

12.7.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

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

Bit 31 **SYNCPOL**: SYNC polarity selection

This bit is set and cleared by software to select the input polarity for the SYNC signal source.

0: SYNC active on rising edge (default)

1: SYNC active on falling edge

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **SYNCSRC[1:0]**: SYNC signal source selection

These bits are set and cleared by software to select the SYNC signal source (see *Section 12.4.2: CRS internal signals*).

00: crs_sync_in_0 selected as SYNC signal source

01: crs_sync_in_1 selected as SYNC signal source

10: crs_sync_in_2 selected as SYNC signal source

11: crs_sync_in_3 selected as SYNC signal source

Note: When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the 48 MHz clock on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **SYNCDIV[2:0]**: SYNC divider

These bits are set and cleared by software to control the division factor of the SYNC signal.

000: SYNC not divided (default)

001: SYNC divided by 2

010: SYNC divided by 4

011: SYNC divided by 8

100: SYNC divided by 16

101: SYNC divided by 32

110: SYNC divided by 64

111: SYNC divided by 128

Bits 23:16 **FELIM[7:0]**: Frequency error limit

FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to *Section 12.4.5* for more details about FECAP evaluation.

Bits 15:0 **RELOAD[15:0]**: Counter reload value

RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to *Section 12.4.4* for more details about counter behavior.

12.7.3 CRS interrupt and status register (CRS_ISR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FECAP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FEDIR	Res.	Res.	Res.	Res.	TRIM OVF	SYNC MISS	SYNC ERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNC WARNF	SYNC OKF
r					r	r	r					r	r	r	r

Bits 31:16 **FECAP[15:0]**: Frequency error capture

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to *Section 12.4.5* for more details about FECAP usage.

Bit 15 **FEDIR**: Frequency error direction

FEDIR is the counting direction of the frequency error counter latched in the time of the last SYNC event. It shows whether the actual frequency is below or above the target.

0: Up-counting direction, the actual frequency is above the target

1: Down-counting direction, the actual frequency is below the target

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **TRIMOVF**: Trimming overflow or underflow

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No trimming error signaled

1: Trimming error signaled

Bit 9 **SYNCMISS**: SYNC missed

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC missed error signaled

1: SYNC missed error signaled

Bit 8 **SYNCERR**: SYNC error

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

0: No SYNC error signaled

1: SYNC error signaled

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **ESYNCF**: Expected SYNC flag

This flag is set by hardware when the frequency error counter reached a zero value. An interrupt is generated if the ESYNCIE bit is set in the CRS_CR register. It is cleared by software by setting the ESYNCC bit in the CRS_ICR register.

0: No expected SYNC signaled

1: Expected SYNC signaled

Bit 2 **ERRF**: Error flag

This flag is set by hardware in case of any synchronization or trimming error. It is the logical OR of the TRIMOVF, SYNCMISS and SYNCERR bits. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software in reaction to setting the ERRC bit in the CRS_ICR register, which clears the TRIMOVF, SYNCMISS and SYNCERR bits.

0: No synchronization or trimming error signaled

1: Synchronization or trimming error signaled

Bit 1 **SYNCWARNF**: SYNC warning flag

This flag is set by hardware when the measured frequency error is greater than or equal to FELIM * 3, but smaller than FELIM * 128. This means that to compensate the frequency error, the TRIM value must be adjusted by two steps or more. An interrupt is generated if the SYNCWARNIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCWARNC bit in the CRS_ICR register.

0: No SYNC warning signaled

1: SYNC warning signaled

Bit 0 **SYNCOKF**: SYNC event OK flag

This flag is set by hardware when the measured frequency error is smaller than FELIM * 3. This means that either no adjustment of the TRIM value is needed or that an adjustment by one trimming step is enough to compensate the frequency error. An interrupt is generated if the SYNCOKIE bit is set in the CRS_CR register. It is cleared by software by setting the SYNCOKC bit in the CRS_ICR register.

0: No SYNC event OK signaled

1: SYNC event OK signaled

12.7.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 3 **ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

Bit 2 **ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.

Bit 1 **SYNCWARNC**: SYNC warning clear flag
Writing 1 to this bit clears the SYNCWARNF flag in the CRS_ISR register.

Bit 0 **SYNCOKC**: SYNC event OK clear flag
Writing 1 to this bit clears the SYNCOKF flag in the CRS_ISR register.

12.7.5 CRS register map

Table 124. CRS register map and reset values

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]						SWSYNC	AUTOTRIMEN	CEN	Res.	ESYNCIE	ERRIE	SYNCWARNIE	SYNCOKIE
	Reset value	0x0000 2000																															
0x04	CRS_CFGR	SYNCPOL	Res.	SYNC SRC [1:0]		Res.	SYNC DIV [2:0]			FELIM[7:0]								RELOAD[15:0]															
	Reset value	0		1	0		0	0	0	0	0	1	0	0	0	1	0	1	0	1	1	1	0	1	1	0	1	1	1	1	1	1	1
0x08	CRS_ISR	FECAP[15:0]																FEDIR	Res.	Res.	Res.	Res.	TRIMOVF	SYNCMISS	SYNCERR	Res.	Res.	Res.	Res.	ESYNCF	ERRF	SYNCWARNF	SYNCOKF
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0					0	0	0	0
0x0C	CRS_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESYNCC	ERRC	SYNCWARNC	SYNCOKC
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

13 General-purpose I/Os (GPIO)

13.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit reset register (GPIOx_BRR), and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL), a secure configuration register (GPIOx_SECCFGR), and a high-speed low-voltage register (GPIOx_HSLVR).

13.2 GPIO main features

- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_ BSRR) for bit-wise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions
- TrustZone® security support
- I/Os state retention during Standby mode

13.3 GPIO functional description

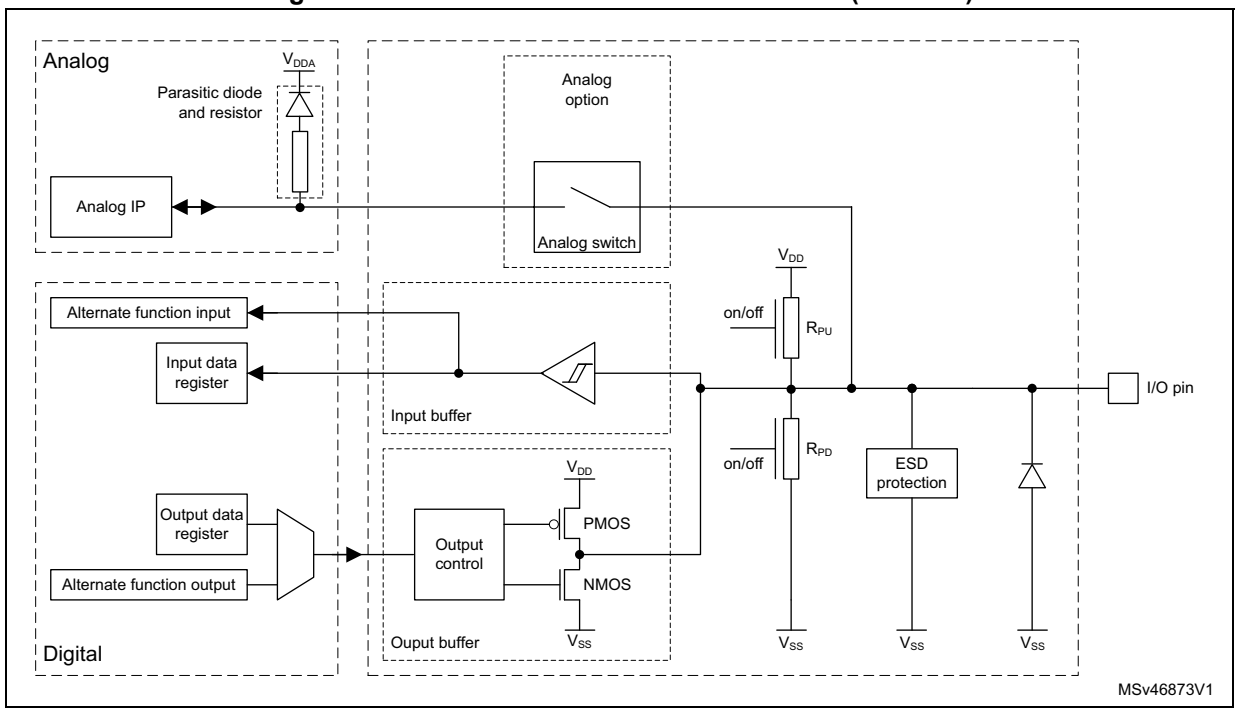
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:

- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words, or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.

Figure 59 shows the basic structure of a 3- or 5-V tolerant GPIO (TT or FT). *Table 125* gives the possible port bit configurations.

Figure 59. Structure of 3- or 5-V tolerant GPIO (TT or FT)



Note: On a TT GPIO, the analog switch is not present, it is replaced by a direct connection. The analog bloc parasitic circuitry does not allow 5-V tolerance.

Table 125. Port bit configuration(1)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]	PUPD(i)[1:0]		I/O configuration	
01	0	SPEED[1:0]	0	0	GP output	PP
	0		0	1	GP output	PP + PU
	0		1	0	GP output	PP + PD
	0		1	1	Reserved	
	1		0	0	GP output	OD
	1		0	1	GP output	OD + PU
	1		1	0	GP output	OD + PD
	1		1	1	Reserved (GP output OD)	

Table 125. Port bit configuration(1) (continued)

MODE(i)[1:0]	OTYPE(i)	OSPEED(i)[1:0]		PUPD(i)[1:0]		I/O configuration	
10	0	SPEED[1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

13.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state no pull-up/pull-down

BOOT0 is in input mode during the reset until at least the end of the option byte loading phase (see *Section 13.3.15*).

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is high-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, which can be activated or not, depending upon the value in the GPIOx_PUPDR register.

13.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input, or analog in the GPIOx_MODER register.
- **Peripheral alternate function**:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down, and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions:**
  - For the ADC and DAC, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC and DAC registers.
  - For the additional functions like RTC, WKUPx, and oscillators, configure the required function in the related RTC, PWR, and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

13.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.

13.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (*GPIO port input data register (GPIOx_IDR) (x = A to I)* and *GPIO port output data register (GPIOx_ODR) (x = A to I)*).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data entered through the I/Os are stored into the input data register (GPIOx_IDR), a read-only register.

13.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

13.3.6 GPIO locking mechanism

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH and GPIOx_HSLVR.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence can be performed only using a word (32-bit long) access to the GPIOx_LCKR register, because GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

13.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that some peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can thus

select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

13.3.8 External interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output, or alternate function mode (the port must not be configured in analog mode). Refer to *Section 18: Extended interrupts and event controller (EXTI)*.

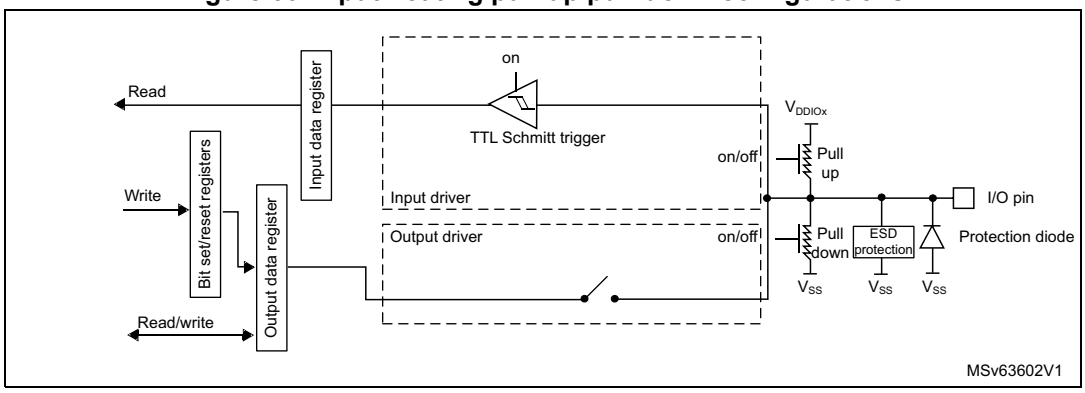
13.3.9 Input configuration

When the I/O port is programmed as input:

- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

Figure 60 shows the input configuration of the I/O port bit.

Figure 60. Input floating/pull-up/pull-down configurations



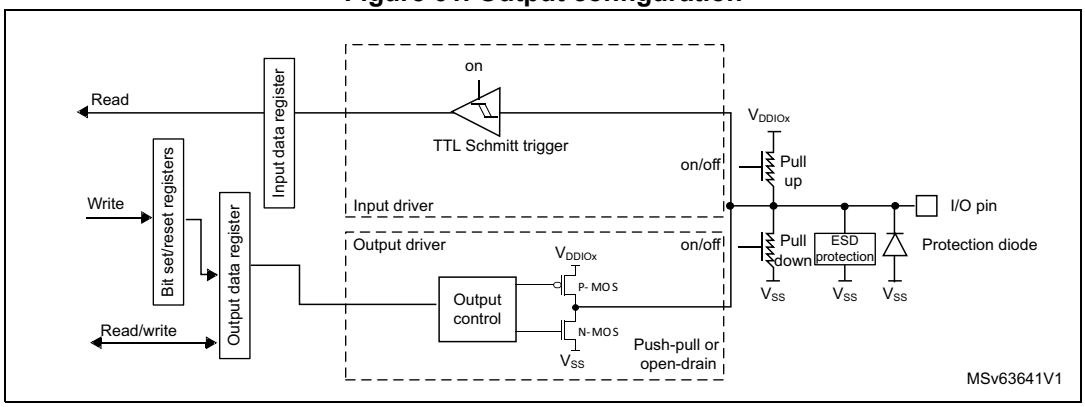
13.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in high-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

Figure 61 shows the output configuration of the I/O port bit.

Figure 61. Output configuration



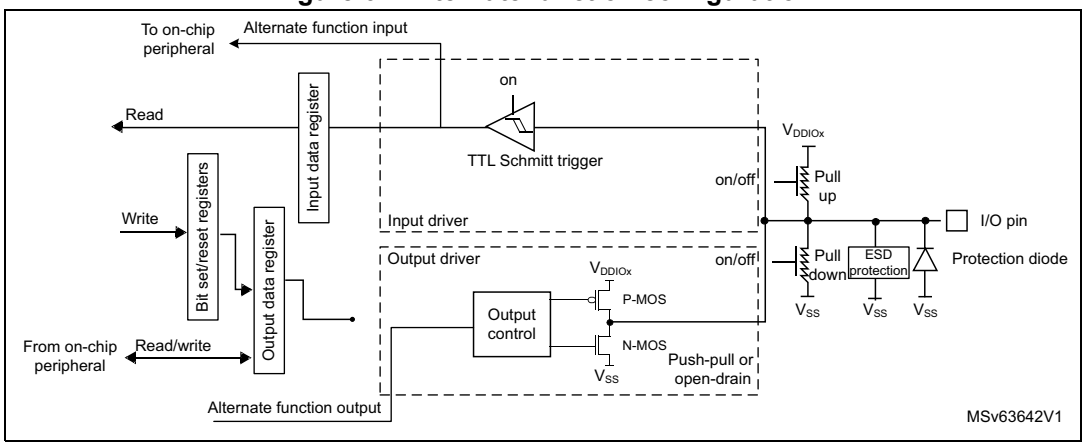
13.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.

Figure 62 shows the alternate function configuration of the I/O port bit.

Figure 62. Alternate function configuration



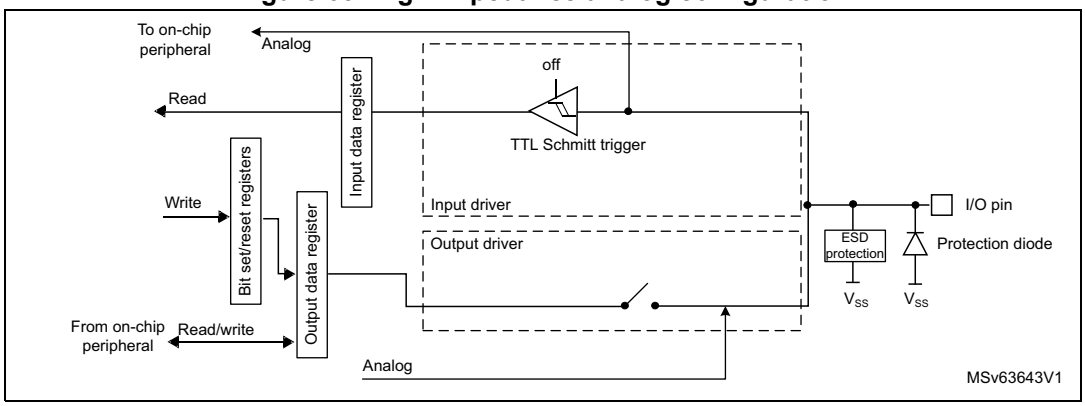
13.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware.
- Read access to the input data register gets the value 0.

Figure 63 shows the high-Z, analog-input configuration of the I/O port bits.

Figure 63. High-impedance analog configuration



13.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in the RCC_CSR register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC_OUT or OSC32_OUT pin can still be used as normal GPIO.

13.3.14 Using the GPIO pins in the RTC supply domain

The PC13/PC14/PC15/PI8 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to *Section 46.3: RTC functional description*

13.3.15 I/Os state retention during standby mode

In the Standby mode, the I/Os are by default in floating state.

If the IORETEN bit in the PWR_IORETR register is set, the I/Os state is sampled during standby entry. The state of I/Os is applied to the pin via pull-up and pull-down resistors. The pull-up and pull-down resistors remains applied after Standby wake-up until the IORETEN bit in the PWR_IORETR register is cleared by software.

13.3.16 TrustZone security

The TrustZone security is activated by the TZEN option byte in the FLASH option byte register. When the TrustZone is active (TZEN = 0xB4), each I/O pin of GPIO port can be individually configured as secure through the GPIOx_SECCFGR register.

When the selected I/O pin is configured as secure, its corresponding configuration bits for alternate function, mode selection, I/O data are secure against a non-secure access. In case of non-secure access, these bits are RAZ/WI.

The I/Os with peripherals functions are also conditioned by the peripheral security configuration (see *Section 5: Global TrustZone® controller (GTZC)* for more details):

- For peripherals for which the I/O pin selection is done through alternate functions registers: if the peripheral is configured as secure, it cannot be connected to a non-secure I/O pin. If this is not respected, the input data to the secure peripheral is forced to 0 (I/O input pin value is ignored) and the output pin value is forced to 0, thus avoiding any secure information leak through non-secure I/Os.
- For I/Os with analog switches, directly controlled by peripherals (such as ADC for instance): If the I/O is secure, the I/O analog switch cannot be controlled by a non-secure peripheral. If this is not respected, the switch remains open. This prevents the redirection of secure data to a non-secure peripheral or I/O through the analog path. Refer to *Section 3: System security* for more details.
- Some of the paths between I/Os “additional functions” and peripherals are not blocked if the I/O is secure and the peripheral is non-secure. Therefore, it is recommended to configure those peripherals as secure even when not used by the application. Refer to *Section 3: System security* for the list of concerned peripherals. When the path has a security control, it follows the same rule as I/O selection through alternate functions.

Refer to the device pins definition table in datasheet for more information about peripherals alternate functions and additional functions mapping.

After reset, all GPIO ports are secure.

The table below gives a summary of the I/O port secured bits following the security configuration bit in the GPIO_SECCFGR register. When the I/O bit port is configured as secure:

- Secured bits: read and write operations are only allowed by a secure access. Non secure-read or write accesses on secured bits are RAZ/WI. There is no illegal access event generated.
- Non-secure bits: no restriction. Read and write operations are allowed by both secure and non-secure accesses.

When the TrustZone security is disabled (TZEN = 0xC3 in FLASH_OPTSR2 register), all registers bits are non-secure. The GPIOx_SECCFGR register is RAZ/WI.

Table 126. GPIO secured bits

Secure configuration bit	Secured bit	Register name	Non-secure access on secure bits
SECy = 1 in GPIOx_SECCFGR(1) (2)	MODEy[1:0]	GPIOx_MODER	RAZ/WI
	OTy	GPIOx_OTYPER	
	OSPEEDy[1:0]	GPIOx_OSPEEDR	
	PUPDy[1:0]	GPIOx_PUPDR	
	IDy	GPIOx_IDR	
	ODy	GPIOx_ODR	
	BSy and BRy	GPIOx_BSRR	
	LCKy	GPIOx_LCKR	
	BRy	GPIOx_BRR	
	AFSELy[3:0]	GPIOx_AFRH	
		GPIOx_AFRL	
	HSLVy	GPIOx_HSLVR	

1. GPIOx, x = A to I. For x = A to H, y = 0 to 15. For x = I, y = 0 to 11.
2. The number of GPIOx ports depends upon the device. Refer to the product datasheet for availability of a particular port. If not present, consider the associated bits as reserved, and keep them at reset value.

13.3.17 Privileged and unprivileged modes

All GPIO registers can be read and written by privileged and unprivileged accesses, whatever the security state (secure or non-secure).

13.3.18 High-speed low-voltage mode (HSLV)

Some I/Os have the capability to increase their maximum speed at low voltage by configuring them in HSLV mode. The I/O HSLV bit controls whether the I/O output speed is optimized to operate at 3.3 V (default setting) or at 1.8 V (HSLV = 1).

Caution: The I/O HSLV configuration bit must not be set if the I/O supply (V_{DD} or V_{DDIO2}) is above 2.7 V. Setting it while the voltage is higher than 2.7 V can damage the device. The I/O HSLV bit can be set only when the corresponding option bit is activated (IO_VDD_HSLV or IO_VDDIO2_HSLV depending on the I/O supply, refer to *Section 7.4: FLASH option bytes*).

There is no hardware protection associated to this feature so it is recommended to use it only as a static configuration for fixed I/O supply.

13.3.19 I/O compensation cell

The I/O commutation slew rate (tfall / trise) can be adapted by software depending on process, voltage and temperatures conditions, to reduce the I/O noise on power supply. Refer to *Section 14: System configuration, boot, and security (SBS)* for more details.

13.4 GPIO registers

This section gives a detailed description of the GPIO registers. Note that the number of GPIOx ports depends upon the device. Refer to the product datasheet for the availability of a particular port. If not present, consider the associated bits as reserved, and keep them at reset value.

The peripheral registers can be written in word, half word, or byte mode.

13.4.1 GPIO port mode register (GPIOx_MODER) (x = A to I)

Address offset:0x00

Reset value: 0xABFF FFFF (for port A)

Reset value: 0xFFFF FEBF (for port B)

Reset value: 0xFFFF FFFF (for ports C..H)

Reset value: 0x00FF FFFF (for port I)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)
These bits are written by software to configure the I/O mode.
00: Input mode
01: General purpose output mode
10: Alternate function mode
11: Analog mode (reset state)

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to I)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to I)

Address offset: 0x08

Reset value: 0x0C00 0000 (for port A)

Reset value: 0x0000 00C0 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very-high speed

Note: Refer to the device datasheet for the frequency specifications, the power supply, and the load conditions for each speed.

The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to I)

Address offset: 0x0C

Reset value: 0x6400 0000 (for port A)

Reset value: 0x0000 0100 (for port B)

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.5 GPIO port input data register (GPIOx_IDR) (x = A to I)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.6 GPIO port output data register (GPIOx_ODR) (x = A to I)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A to I).

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to I)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 **BSy:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit

1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to I)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

- LOCK key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

- LOCK key read

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the LOCK.

After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 **LCKy:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0

0: Port configuration not locked

1: Port configuration locked

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to I)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 7 to 0)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to H)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** Alternate function selection for port x I/O pin y (y = 15 to 8)
These bits are written by software to configure alternate function I/Os.
0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to I)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port x reset IO pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.
0: No action on the corresponding ODy bit
1: Reset the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.12 GPIO high-speed low-voltage register (GPIOx_HSLVR) (x = A to I)

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HSLVy:** Port x high-speed low-voltage configuration (y = 15 to 0)

These bits are written by software to optimize the I/O speed when the I/O supply is low.

Each bit is active only if the corresponding IO_VDD_HSLV/IO_VDDIO2_HSLV user option bit is set. It must be used only if the I/O supply voltage is below 2.7 V.

Setting these bits when the I/O supply (VDD or VDDIO2) is higher than 2.7 V may be destructive.

0: I/O speed optimization disabled

1: I/O speed optimization enabled

Note: Not all I/Os support the HSLV mode. Refer to the I/O structure in the corresponding datasheet for the list of I/Os supporting this feature. Other I/Os HSLV configuration must be kept at reset value.

The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.13 GPIO secure configuration register (GPIOx_SECCFGR) (x = A to I)

When the system is secure (TZEN = 0xB4), this register provides write access security and can be written only by a secure access. It is used to configure a selected I/O as secure. A non-secure write access to this register is discarded.

When the system is not secure (TZEN = 0xC3), this register is RAZ/WI.

Address offset: 0x30

Reset value: 0x0000 FFFF (for ports A to H)

Reset value: 0x0000 0FFF (for port I)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECy**: I/O pin of Port x secure bit enable y (y = 15 to 0)

These bits are written by software to enable or disable the I/O port pin security.

0: The I/O pin is non-secure

1: The I/O pin is secure. Refer to *Table 126* for all corresponding secured bits.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

13.4.14 GPIO register map

Table 127. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**GPIOx_MODER (x = A to I)**	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value for port A	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
	Reset value for ports C...H	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port I	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	**GPIOx_OTYPER (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**GPIOx_ OSPEEDR (x = A to I)**	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value for port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	Reset value for ports C...I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**GPIOx_PUPDR (x = A to I)**	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value for port A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value for port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value for ports C...I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**GPIOx_IDR (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	**GPIOx_ODR (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**GPIOx_BSRR (x = A to I)**	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**GPIOx_LCKR (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**GPIOx_AFRL (x = A to I)**	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	**GPIOx_AFRH (x = A to H)**	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 127. GPIO register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**GPIOx_BRR (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**GPIOx_HSLVR (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSLV15	HSLV14	HSLV13	HSLV12	HSLV11	HSLV10	HSLV9	HSLV8	HSLV7	HSLV6	HSLV5	HSLV4	HSLV3	HSLV2	HSLV1	HSLV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	**GPIOx_ SECCFGR (x = A to I)**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value for A to H																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value for port I																	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3* for the register boundary addresses.

14 System configuration, boot, and security (SBS)

14.1 SBS introduction

The devices feature a set of configuration registers located in the SBS. On top of various device configurations, this peripheral controls key boot and security features, including debug control and secure storage control.

14.2 SBS main features

- System configuration
  - Manage safety feature
  - Enable/disable the FMP (Fast-mode Plus) high-drive capability of some I/Os and voltage booster for I/O analog switches
  - Manage the I/O compensation cell
  - Configure register security access
- Boot control
  - Upon system reset, configure the Cortex-M33 boot address and the temporal isolation level depending on the current configuration such as PRODUCT_STATE (sbs_product_state), BOOT_UBE (sbs_irot_select), or TZEN (sbs_tzen).
  - Manage the temporal isolation feature implemented thanks to the hide protect level (HDPL) monotonic counter
- Debug control
  - Control the opening of the device debug interface, ensuring the sequencing of events that guarantee the device security
- Hardware secure storage control
  - Control the secure storage selections: OBK-HDPL selection (selects the area of secure storage of the flash memory and selects the corresponding DHUK in SAES), EPOCH selection

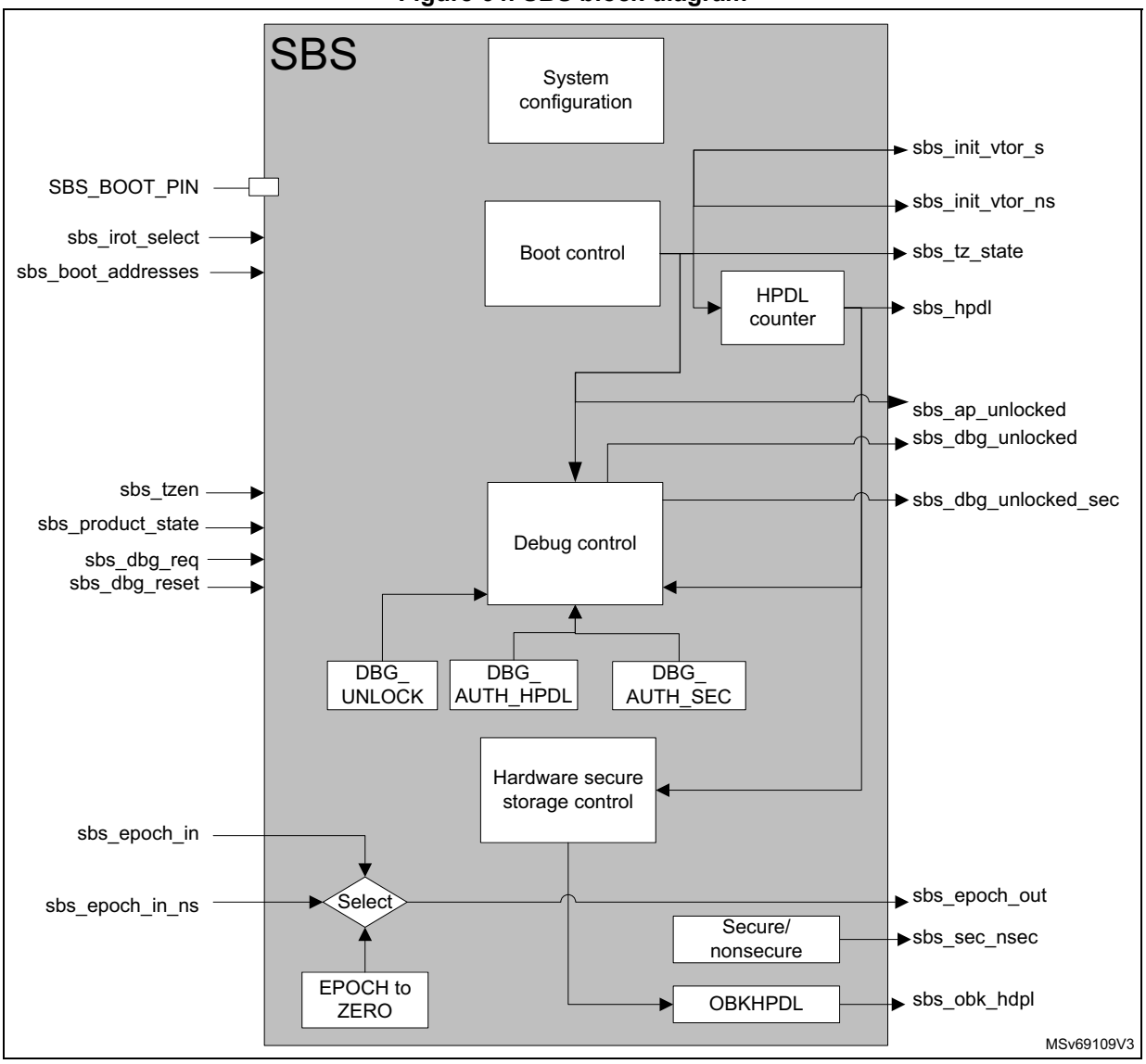
14.3 SBS functional description

14.3.1 SBS block diagram

Figure 64 shows the SBS block diagram, including the four main functions:

- System configuration
- Boot control
- Debug control
- Hardware secure storage control

Figure 64. SBS block diagram



14.3.2 SBS signals

Table 128 details the user relevant internal signals that interface the SBS.

Table 128. SBS internal input/output signals

Signal name	Type	Description
BOOT0	Input	Select booting on user flash memory or Bootloader when TrustZone is disabled (TZEN = 0xC3), or on RSS when TrustZone is enabled (TZEN = 0xB4).
sbs_irot_select		Signal based on BOOT_UBE option byte to select the iROT between ST-iROT and user flash memory (OEM-iROT)
sbs_tzen		Signal based on TZEN option byte to activate/deactivate the TrustZone
sbs_boot_addresses		List of addresses defined by the flash memory: – NSBOOTADD: nonsecure boot address – SECBOOTADD: secure boot address
sbs_product_state		Signal based on PRODUCT_STATE option byte to activate the different security mechanisms depending on the product use. Expected values are described in *Section 7: Embedded flash memory (FLASH)*.
sbs_dbg_req		Launch the debug authentication protocol when booting.
sbs_init_vtor_s	Output	Vector address for Cortex-M33 secure entry point
sbs_init_vtor_ns		Vector address for Cortex-M33 nonsecure entry point
sbs_tz_state		Inform the Cortex-M33 on the secure state of the core.
sbs_hdpl		HDPL (temporal isolation level, ID of the boot level, used to isolate boot levels) This signal reflects a monotonic counter that can be incremented from 0 to 3, and that is reset to zero only on software reset or POR.
sbs_ap_unlocked		Control the Cortex-M33 access port.
sbs_dbg_unlocked		Unlock the debug (when set to 1) for the Cortex-M33 nonsecure part.
sbs_dbg_unlocked_sec		Unlock the debug (when set to 1) for the Cortex-M33 secure part.
sbs_dbg_reset	Input	This signal is used to control the reset of the debug authentication configuration to be done with a system reset or a power-on reset. The configuration is done through the DBGMCU using the DCRT bitfield of DBGMCU_CR register.
sbs_obk_hdpl	Output	Select secure storage domain (OBK-HDPL) for current HDPL, or greater ones (to allow provisioning).
sbs_epoch_out		EPOCH counters (NS_EPOCH and SEC_EPOCH, inputs from flash memory) are used to manage the REPLAY protection.
sbs_sec_nsec		Reflect secure/nonsecure selection for the secure storage.

14.3.3 SBS reset and clocks

The SBS configuration port is clocked by the AHB bus clock. There is a general reset and a debug configuration reset controlled in DBGMCU.

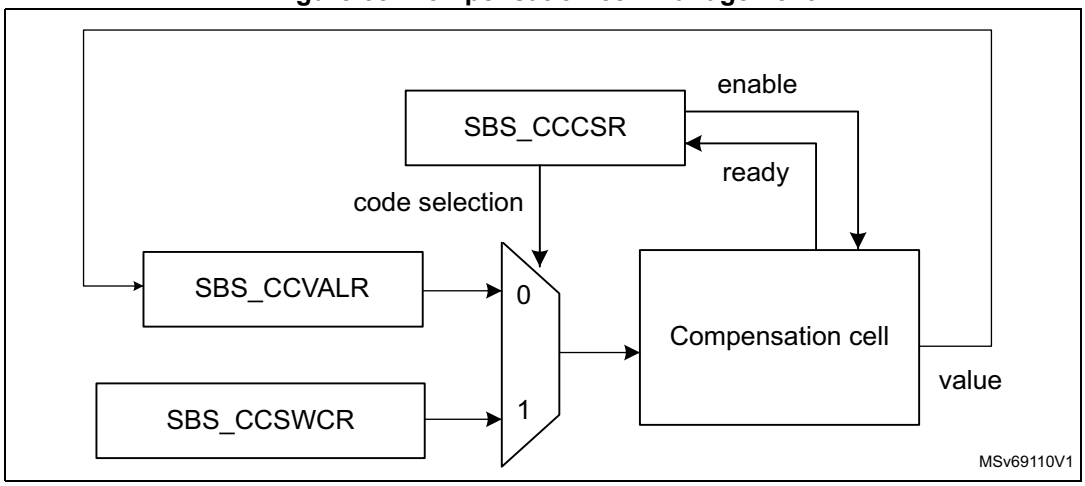
14.3.4 SBS system configuration

SBS I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the output impedance in the I/O buffer, and the slew rate of the I/O commutation (t_{fall} and t_{rise}), to reduce the I/O noise on power supply.

As shown in *Figure 65*, the compensation cell is split in two blocks, one to provide an optimal code for the current PVT, and another to drive the block controlled by the software.

Figure 65. Compensation cell management



The compensation cell value can be read when the READY flag is set in SBS_CCCSR. With CODESEL in SBS_CCCSR, the application can select the value to apply between two options: the code from the cell or the code from SBS_CCSWCR.

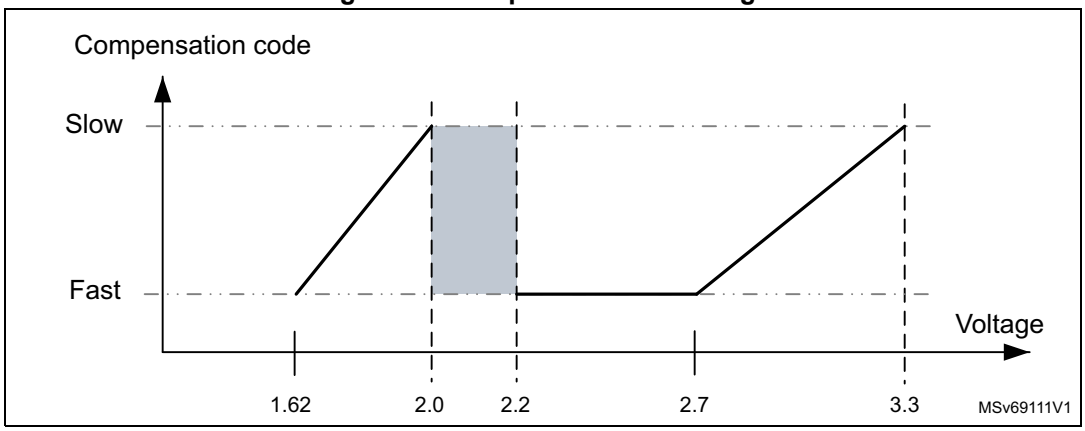
Two compensation cells are embedded in STM32H5 devices, one for the I/Os supplied by V_{DDIO} power rail, another for the I/Os supplied by V_{DDIO2} power rail.

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os.

Note: The compensation cell can be used only when 2.7 V ≤ V_{DDIOx} ≤ 3.6 V or 1.62 V ≤ V_{DDIOx} ≤ 2 V (see Figure 66).

Note: The compensation cell can be used only when the CSI oscillator is enabled, see Section 11: Reset and clock control (RCC) for more details on CSI oscillator.

Figure 66. Compensation cell usage



SBS TrustZone security and privilege

- SBS TrustZone security

  When the TrustZone security is activated, the SBS is able to protect secure registers from being modified by nonsecure accesses.

  The TrustZone security is activated by the TZEN option byte in FLASH_OPTSR2_PRG.

  A nonsecure read/write access to a secured register is RAZ/WI and generates an illegal access event. An illegal access interrupt is generated if the SBS illegal access event is enabled in the GTZC.
- Privileged/unprivileged mode

  The SBS registers can be read and written by privileged and unprivileged accesses except the SBS registers for CPU configuration:

  – SBS_CSLCKR, SBS_FPUIMR and SBS_CNSLCKR

  – FPUSEC in SBS_SECCFGR

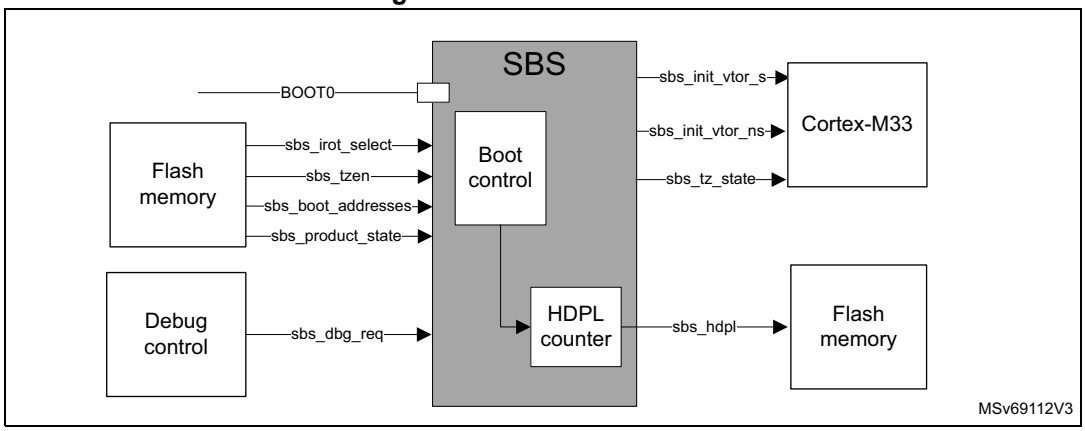
  An unprivileged access to a privileged register is RAZ/WI.

14.3.5 SBS boot control

The SBS can be used to control the boot entry points considering the product settings. The main boot control actions are listed below:

- Run product with or without TrustZone enabled.
- Select between ST-iROT or OEM-iROT.
- Boot when launching a debug authentication sequence.
- Select boot between the bootloader or the user flash memory boot.
- Initialize the HDPL boot value.

Figure 67. SBS boot control



The boot configurations are selected considering the product settings:

- BOOT0: to select booting on user flash memory or RSS (root secure services)
- BOOT_UBE option byte to select the iROT between ST-iROT and OEM-iROT
- TZEN option byte to activate/deactivate the Trust Zone
- sbs_boot_addresses: list of addresses defined by the flash memory:
  - NSBOOTADD: nonsecure boot address
  - SECBOOTADD: secure boot address
- PRODUCT_STATE: option byte to activate the different security mechanisms depending on the product use. Expected values are described in *Section 7: Embedded flash memory (FLASH)*.
- sbs_dbg_req: used to launch the debug authentication protocol when booting

The boot control logic sets the following data:

- sbs_init_vtor_s: vector address for Cortex-M33 secure entry point
- sbs_init_vtor_ns: vector address for Cortex-M33 nonsecure entry point
- sbs_tz_state (secure/nonsecure): informs the Cortex-M33 on the secure state of the core.
- HDPL (see the description below)

SBS HDPL (temporal isolation level) management

The HDPL is a monotonic counter incremented during the boot stages. The HDPL is reset to its default value only after a power-on or a system reset. This default value (0 or 1) depends on the device life cycle, as defined in boot logic.

The devices use HDPL information to automatically isolate code and its associated secrets (like keys) during the boot process. Incrementing HPDL ensures that private code and data for one boot stage cannot be directly accessible from later boot stages.

The HDPL is used by the user flash memory, see *Section 7: Embedded flash memory (FLASH)* for more details. The HDPL can take values from 0 to 3. When reaching 3, HDPL keeps this value until reset. The current HDPL value is readable in HDPL bitfield in SBS_HDPLSR.

To increment the HDPL by one, the application must write 0x6A to INCR_HDPL in SBS_HDPLCR. After such increment, and before doing any subsequent action, the user must check that the HDPL has effectively been incremented, by reading SBS_HDPLSR.

Table 129. HDPL encoded values

HDPL	Code
0	0xB4
1	0x51
2	0x8A
3	0x6F
	All other values

Table 130. SBS boot logic

Inputs						Outputs			
sbs_product_state	**sbs_dbg_req**	**sbs_tzen**	**sbs_irotselect**	**BOOT0**	**sbs_boot_addresses**	**sbs_init_vtor_s**	**sbs_init_vtor_ns**	**sbs_hdpl**	**sbs_tz_state**
Any except Locked	1	x	x	x	BOOT_DBG_AUTH_ADD	BOOT_DBG_AUTH_ADD	x	1	Secure
Open	0	0	x	0	NSBOOT ADD	x	NSBOOT ADD	1	Nonsecure
	0	0	x	1	BOOT_ST_RSS_ADD	BOOT_ST_RSS_ADD	x	0	Secure
	0	1	x	0	SECBOOT ADD	SECBOOT ADD	x	1	secure
	0	1	x	1	BOOT_ST_RSS_ADD	BOOT_ST_RSS_ADD	x	0	Secure
Provisioning	0	x	x	x	BOOT_ST_RSS_ADD	BOOT_ST_RSS_ADD	x	0	Secure
iROT-Provisioned	0	0	x	x	NSBOOT ADD	x	NSBOOT ADD	1	Nonsecure
	0	1	0	x	BOOT_ST_iROT_ADD	BOOT_ST_iROT_ADD	x	1	Secure
	0	1	1	x	SECBOOT ADD	x	SECBOOT ADD	1	Secure
TZ-closed	-	1	0	x	BOOT_ST_iROT_ADD	BOOT_ST_iROT_ADD	x	1	Secure
	-	1	1	x	SECBOOT ADD	x	SECBOOT ADD	1	Secure

Table 130. SBS boot logic (continued)

Inputs						Outputs			
sbs_product_state	**sbs_dbg_req**	**sbs_tzen**	**sbs_irotselect**	**BOOT0**	**sbs_boot_addresses**	**sbs_init_vtor_s**	**sbs_init_vtor_ns**	**sbs_hdpl**	**sbs_tz_state**
Closed	0	0	x	x	NSBOOT ADD	x	NSBOOT ADD	1	Nonsecure
	0	1	0	x	BOOT_ST_iROT_ADD	BOOT_ST_iROT_ADD	x	1	Secure
	0	1	1	x	SECBOOT ADD	x	SECBOOT ADD	1	Secure
Locked	0	0	x	x	NSBOOT ADD	x	NSBOOT ADD	1	Nonsecure
	0	1	0	x	BOOT_ST_iROT_ADD	BOOT_ST_iROT_ADD	x	1	Secure
	0	1	1	x	SECBOOT ADD	x	SECBOOT ADD	1	Secure
Regression	0	0	x	x	BOOT_DBG_AUTH_ADD	BOOT_DBG_AUTH_ADD	x	0	Secure
	0	1	x	x	BOOT_DBG_AUTH_ADD	BOOT_DBG_AUTH_ADD	x	0	Secure
NS-Regression	0	1	x	x	BOOT_DBG_AUTH_ADD	BOOT_DBG_AUTH_ADD	x	0	Secure

14.3.6 SBS debug control

The SBS debug control is used to manage debug opening, taking care on the product context (PRODUCT_STATE, TZEN, HDPL) on register settings, or through a debug authentication control.

When the debug is forbidden, the mailbox access port, Cortex-M33 access port and CPU debug interface are locked. In this situation, the debugger cannot access the CPU and no effective debug can be done. Refer to the *Section 59: Debug support (DBG)* for more details.

Authenticated debug sequence

1. The external host requests to launch the debug authentication protocol, via the DBGMCU access port mailbox. The rest of the device is kept under reset.
2. SBS selects the STMicroelectronics RSS-DA (debug authentication library) boot address, and requests the CPU to be released from reset.

3. The CPU running RSS-DA library executes the debug authentication protocol in the system flash memory. If the device is closed, the access port mailbox is closed until RSS-DA acknowledges the authentication sequence start request.
4. The authentication method depends on TrustZone activation:
   – When TrustZone is activated (TZEN = 0xB4), the authentication method is based on certificates. As soon as a debug certificate chain is fully verified by the device, if the certificate concerns a debug permission, the RSS-DA programs the debug opening of the Cortex-M33. Alternatively, the certificate can authorize partial or full regression, allowing debug on a regressed part.
   – When TrustZone is disabled (TZEN = 0xC3), the authentication method is based on password. This method only allows the full regression of the product to be controlled.
5. Above reopenings are effective only when HDPL in SBS_HDPLSR has a value equal or superior to the value programmed in DBG_AUTH_HDPL in SBS_DBGCR. In case of authentication failure, the user is informed through the host interface.

*Note: The debug authentication library in system flash memory is available only when HDPL = 0 or 1 in SBS_*HDPL*SR. Only this library can perform the steps 3 and 4 described above.*

Debug reset

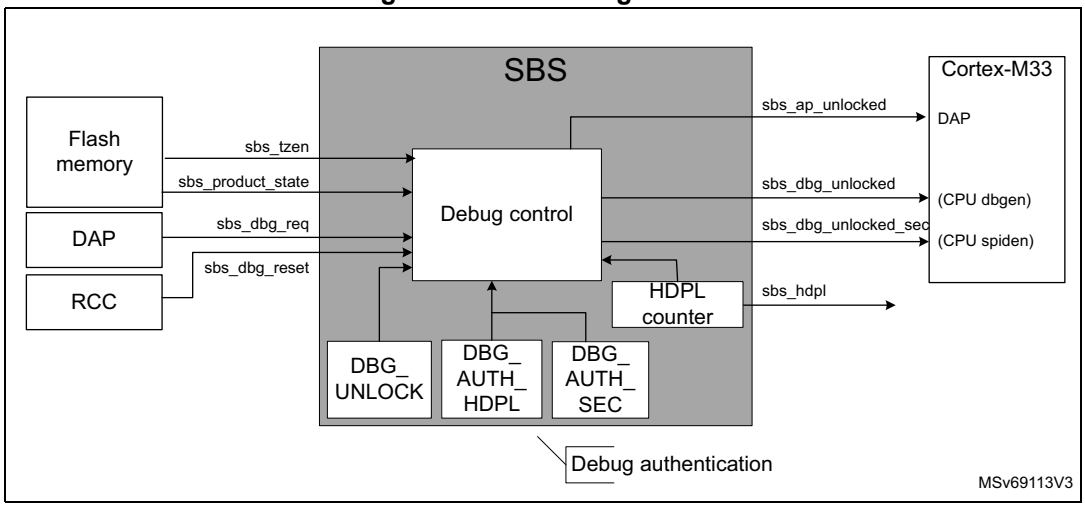
The debug opening can have the configuration to be reset by system or power-on reset, depending on a DBGMCU register field.

Debug locking

The debug configuration can be locked thanks to DBGCFG_LOCK in SBS_DBGLOCKR. SBS_DBGCR is then no longer writable.

When DBGCFG_LOCK is set to 1, it can be reset only by system or power-on reset. The configuration is done through the DBGMCU using the DCRT field of DBGMCU_CR.

Figure 68. SBS debug control



Inputs used to control the debug opening:

- sbs_dbg_req: input signal that a host requests to launch the debug authentication protocol. When this signal is set, the boot address is set to launch the debug authentication library.
- sbs_tzen: inform on the selected platform configuration related to TrustZone activation. When TZEN value is set to zero (0xC3), the platform does not support TrustZone and the simplified life cycle is proposed. See *Section 7: Embedded flash memory (FLASH)* for more details.
- sbs_product_state: provide the current PRODUCT_STATE (from flash memory). See *Section 7: Embedded flash memory (FLASH)* for more details.
- sbs_dbg_reset: reset information coming from the RCC, and reset SBS_DBGCR if this register is configured to be reset.

Configuration

- DBG_UNLOCK in SBS_DBGCR: debug unlock when DBG_AUTH_HDPL is reached
- AP_UNLOCK in SBS_DBGCR: access port unlock
- DBG_AUTH_HDPL in SBS_DBGCR: authenticated debug temporal isolation level. Define the HDPL value from which the debug can be opened. Value can only be from 1 to 3.
- DBG_AUTH_SEC in SBS_DBGCR: specify if the debug reopening is for nonsecure only or for both (secure and nonsecure).
- DBGC_LOCK in SBS_DBGLOCKR: lock the current debug configuration (released only by a reset).

Outputs

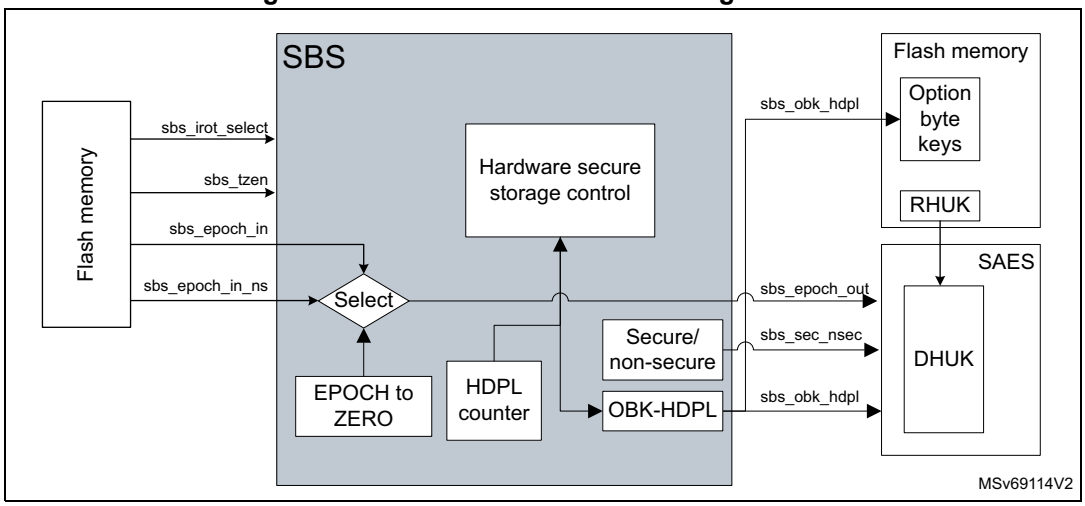
- sbs_ap_unlocked: signal controlling the Cortex-M33 access port
  - 1: 0xB4
  - 0: all other codes
- sbs_dbg_unlocked: unlock the debug (when set to 1) for the Cortex-M33 nonsecure.
  - 1: 0xB4
  - 0: all other codes
- sbs_dbg_unlocked_sec: unlock the debug (when set to 1) for the Cortex-M33 secure.
  - 1: 0xB4
  - 0: all other codes

14.3.7 SBS hardware secure storage control

This feature ensures the isolation of keys and data related to ROT (root-of-trust) when re-opening the debug (when product in the field).

This includes a dedicated area called OB-Keys in the flash memory (see *Section 7: Embedded flash memory (FLASH)* for more details), and the key derivation (DHUK) mechanism protecting data using a hardware key different for the identified domains.

Figure 69. SBS hardware secure storage control



- sbs_obk_hdpl: select secure storage context for current HDPL, or greater ones (to allow provisioning). This signal is set thanks to the SBS_NEXTHDPLCR register. This information is used by the flash memory to select the right area, and by the SAES when processing the derived key to generate a key derived related to HDPL context.
- sbs_epoch_out: EPOCH counters are used to manage the REPLAY protection. These counters are incremented when regressions are done. Injecting the EPOCH in the DHUK (in the SAES) ensures that all information encrypted with the DHUK cannot be replayed after a regression.
- sbs_sec_nsec: apply a different protection DHUK for secure and nonsecure assets.

All keys encrypted using the SAES/DHUK inherit of the RHUK property: unique per device.

All data encrypted thanks to the SAES using the DHUK are specific to a combination of [sbs_obk_hdpl + sbs_epoch_out + sbs_sec_nsec].

Inputs used to control the hardware secure storage control

sbs_epoch_in and sbs_epoch_in_ns: 24-bit values coming from the flash memory and representing regression counters respectively for secure and nonsecure.

Configuration

- NEXTHDPL in SBS_NEXTHDPLCR: access to next HDPL secure storage areas.
- EPOCH_SEL in SBS_EPOCHSELCR: select the EPOCH source related to the secure storage under control (SEC_EPOCH, NS_EPOCH, or FORCED_To_Zero).

Table 131. OBK-HDPL logic

HDPL[7:0] in SBS_HDPLSR	NEXTHDPL[1:0]			
	0x0	0x1	0x2	0x3
0 (0xB4)	0xB4	0x51	0x8A	0x6F
1 (0x51)	0x51	0x8A	0x6F	
2 (0x8A)	0x8A	0x6F		
3 (0x6F)	0x6F			
Others				

14.4 SBS interrupts

SBS does not support interrupts.

14.5 SBS registers

14.5.1 SBS temporal isolation control register (SBS_HDPLCR)

Address offset: 0x010

Reset value: 0x0000 00B4

Reset: system reset

Register security: no restriction

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

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

Bits 7:0 **INCR_HDPL[7:0]**: increment HDPL value
0xB4: no increment
0x6A: recommended value to increment HDPL level by one
Others: all other values allow a HDPL level increment.

14.5.2 SBS temporal isolation status register (SBS_HDPLSR)

Address offset: 0x014

Reset value: 0xXXXX XXXX

The reset value depends on boot case: booting with HDPL0 for ST code, or HDPL1 for all other cases. See *Table 130* for more details.

Reset: system reset

Register security: no restriction

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

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

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

Bits 7:0 **HDPL[7:0]**: temporal isolation level

This bitfield returns the current temporal isolation level.
0xB4: HDPL0, RSS
0x51: HDPL1, iRoT
0x8A: HDPL2, uRoT
0x6F: HDPL3, application (secure/nonsecure)

14.5.3 SBS next HDPL control register (SBS_NEXTHDPLCR)

Address offset: 0x018

Reset value: 0x0000 0000

Reset: system reset

Register security: no restriction

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

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

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

Bits 1:0 **NEXTHDPL[1:0]**: index to point to a higher HDPL than the current one

Index to add to the current HDPL to point (through OBK-HDPL) to the next secure storage areas (OBK-HDPL = HDPL + NEXTHDPL). See *Table 131: OBK-HDPL logic* for more details.

14.5.4 SBS debug control register (SBS_DBGCR)

Address offset: 0x020

Reset value: 0x0000 0000

Reset: debug reset (system reset or power-on reset)

Register security: HDPL0/1, RAZ/WI otherwise

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

Bits 31:24 **DBG_AUTH_SEC[7:0]**: control debug opening secure/nonsecure
Write 0xB4 to this bitfield to open debug for secure and any other values only opens nonsecure

Bits 23:16 **DBG_AUTH_HDPL[7:0]**: authenticated debug temporal isolation level
Writing to this bitfield defines at which HDPL the authenticated debug opens.
0x51: HDPL1
0x8A: HDPL2
0x6F: HDPL3

Note: Writing any other values is ignored. Reading any other value means the debug never opens.

Bits 15:8 **DBG_UNLOCK[7:0]**: debug unlock when DBG_AUTH_HDPL is reached
Write 0xB4 to this bitfield to open the debug when HDPL in SBS_HDPLSR equals to DBG_AUTH_HDPL in this register.

Bits 7:0 **AP_UNLOCK[7:0]**: access port unlock
Write 0xB4 to this bitfield to open the device access port.

14.5.5 SBS debug lock register (SBS_DBGLOCKR)

Address offset: 0x024

Reset value: 0x0000 00B4

Reset: debug reset (system reset or power-on reset)

Register security: HDPL0/1, RAZ/WI otherwise

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

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

Bits 7:0 **DBGCFG_LOCK[7:0]**: debug configuration lock
Reading this bitfield returns 0x6A if the bitfield value is different from 0xB4.
0xC3 is the recommended value to lock the debug configuration using this bitfield.
0xB4: Writes to SBS_DBGCR allowed (default)
Others: Writes to SBS_DBGCR ignored

14.5.6 SBS RSS command register (SBS_RSSCMDR)

Address offset: 0x034

Reset value: 0x0000 0000

Reset: power-on reset

Register security: always secure (RAZ/WI if nonsecure)

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RSSCMD[15:0]**: RSS command

The application can use this bitfield to pass on a command to the RSS, executed at the next reset.

When RSSCMD ≠ 0 and PRODUCT_STATE is in Open, then the system always boots on RSS whatever is the boot pin value.

14.5.7 SBS EPOCH selection control register (SBS_EPOCHSELCR)

Address offset: 0x0A0

Reset value: 0x0000 0000

Reset: system reset

Register security: Secure when TZ_STATE = 1 (RAZ, WI in nonsecure access). Nonsecure protection when TZ_STATE = 0. This register is protected by privileged whatever is TZ_STATE, RAZ/WI if non privileged access.

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

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

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

Bits 1:0 **EPOCH_SEL[1:0]**: select EPOCH value to be sent to the SAES

00: NS_EPOCH (nonsecure) counter input selected

01: SEC_EPOCH counter input selected

1x: EPOCH forced to zero (value used to retrieve PUF reference value at boot time)

14.5.8 SBS security mode configuration control register (SBS_SECCFGR)

Address offset: 0x0C0

Reset value: 0x0000 0000

Reset: system reset

Register security: always secure. RAZ/WI if nonsecure transaction and TZ_STATE = 1. RAZ/WI if TZ_STATE = 0.

This register is programmed by secure software if the user wants functions configurable through system registers to be secure or not.

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

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

Bit 3 **FPUSEC**: FPU security enable
0: SBS_FPUIMP register accessible through secure or nonsecure transaction
1: SBS_FPUIMP register accessible only through secure transaction
Note: This bit can be written only through privilege transaction.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CLASSBSEC**: ClassB security enable
0: SBS_CFGR2 register accessible through secure or nonsecure transaction
1: SBS_CFGR2 register accessible only through secure transaction

Bit 0 **SBSSEC**: SBS clock control, memory-erase status register and compensation cell register security enable
0: SBS_MESR, SBS_CCCSR, SBS_CCVALR, SBS_CCSWCR registers accessible through secure or nonsecure transaction
1: SBS_MESR, SBS_CCCSR, SBS_CCVALR, SBS_CCSWCR registers accessible only through secure transaction

14.5.9 SBS product mode and configuration register (SBS_PMCR)

Address offset: 0x100

Reset value: 0x0000 0000

Reset: system reset

Register security:

- When TrustZone is activated (TZEN = 0xB4):
  - depending on I/O PBx_SEC_EN secure input for PBx_FMP configuration bits
  - depending on Ethernet ETHSEC secure input for ETH_SEL_PHY configuration bits
- When TrustZone is disabled (TZEN = 0xC3) there is no access restriction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH_SEL_PHY[2:0]			Res.	PB9_FMP	PB8_FMP	PB7_FMP	PB6_FMP
								rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:21 **ETH_SEL_PHY[2:0]**: Ethernet PHY interface selection
000: GMII or MII
001: reserved (RGMII)
100: RMII
Others: reserved
Refer to the product datasheet for the availability of Ethernet. If not present, consider the associated bits as reserved, and keep them at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **PB9_FMP**: Fast-mode Plus driving capability activation on PB9
This bit can be read and written only with secure access if PB9 is secure in GPIOB. This bit enables the Fm+ driving mode for PB9 when PB9 is not used by I2C peripheral. This can be used to drive a LED for instance.
0: PB9 pin operates in standard mode.
1: Fm+ mode is enabled on PB9 pin and the speed control is bypassed.

Bit 18 **PB8_FMP**: Fast-mode Plus driving capability activation on PB8
This bit can be read and written only with secure access if PB8 is secure in GPIOB. This bit enables the Fm+ driving mode for PB8 when PB8 is not used by I2C peripheral. This can be used to drive a LED for instance.
0: PB8 pin operates in standard mode.
1: Fm+ mode is enabled on PB8 pin and the speed control is bypassed.

Bit 17 **PB7_FMP**: Fast-mode Plus driving capability activation on PB7
This bit can be read and written only with secure access if PB7 is secure in GPIOB. This bit enables the Fm+ driving mode for PB7 when PB7 is not used by I2C peripheral. This can be used to drive a LED for instance.
0: PB7 pin operates in standard mode.
1: Fm+ mode is enabled on PB7 pin and the speed control is bypassed.

Bit 16 **PB6_FMP**: Fast-mode Plus driving capability activation on PB6
This bit can be read and written only with secure access if PB6 is secure in GPIOB. This bit enables the Fm+ driving mode for PB6 when PB6 is not used by I2C peripheral. This can be used to drive a LED for instance.
0: PB6 pin operates in standard mode.
1: Fm+ mode is enabled on PB6 pin and the speed control is bypassed.

Bits 15:8 Reserved, must be kept at reset value.

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

14.5.10 SBS FPU interrupt mask register (SBS_FPUIMR)

Address offset: 0x104

Reset value: 0x0000 001F

Reset: system reset

Register security: depends on FPUSEC in SBS_SECCFGR

This register is accessible only through privilege transaction.

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

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

Bits 5:0 **FPU_IE[5:0]**: FPU interrupt enable

Set and cleared by software to enable the Cortex-M33 FPU interrupts
FPU_IE[5]: inexact interrupt enable (interrupt disabled at reset)
FPU_IE[4]: input abnormal interrupt enable
FPU_IE[3]: overflow interrupt enable
FPU_IE[2]: underflow interrupt enable
FPU_IE[1]: divide-by-zero interrupt enable
FPU_IE[0]: invalid operation interrupt enable

14.5.11 SBS memory erase status register (SBS_MESR)

Address offset: 0x108

Reset value: 0x0000 000X (bit 0 is not affected by system reset)

Register security: depends on SBSSEC in SBS_SECCFGR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE
															rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
															rc_w1

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IPMEE**: ICACHE erase status
This bit is set by hardware when ICACHE and PKA RAMs erase is completed after potential tamper detection or a product state regression (refer to *Section 47: Tamper and backup registers (TAMP)* for more details).
This bit is cleared by software by writing 1 to it.
0: ICACHE and PKA RAM erase on going
1: ICACHE and PKA SRAM erase done

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MCLR**: device memories erase status
This bit is set by hardware when SRAM2, BKPSRAM, ICACHE, DCACHE and PKA RAMs erase is completed after power-on reset or tamper detection or product state regression (refer to *Section 47: Tamper and backup registers (TAMP)*).
This bit is not reset by system reset and is cleared by software by writing 1 to it.
0: memory erase on going if not yet cleared by software
1: Memory erase done

14.5.12 SBS compensation cell for I/Os control and status register (SBS_CCCSR)

Address offset: 0x110

Reset value: 0x0000 0000

Reset: system reset

Register security: depends on SBSSEC in SBS_SECCFGR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
						r	r					rw	rw	rw	rw

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

Bit 9 **RDY2**: VDDIO2 compensation cell ready flag
This bit provides the status of the VDDIO2 compensation cell.
0: VDDIO2 compensation cell not ready
1: VDDIO2 compensation cell ready (code value provided by the cell can be used)

Bit 8 **RDY1**: VDDIO compensation cell ready flag
This bit provides the status of the compensation cell.
0: VDDIO compensation cell not ready
1: VDDIO compensation cell ready (code value provided by the cell can be used)

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CS2**: code selection for VDDIO2 power rail (reset value set to 1)
This bit selects the code to be applied for the I/O compensation cell.
0: Code from the cell (available in SBS_CCVR)
1: Code from SBS_CCCR

Bit 2 **EN2**: enable compensation cell for VDDIO2 power rail
This bit enables the I/O compensation cell.
0: I/O compensation cell disabled
1: I/O compensation cell enabled

Bit 1 **CS1**: code selection for VDDIO power rail (reset value set to 1)
This bit selects the code to be applied for the I/O compensation cell.
0: Code from the cell (available in the SBS_CCVR)
1: Code from SBS_CCCR

Bit 0 **EN**1: enable compensation cell for VDDIO power rail
This bit enables the I/O compensation cell.
0: I/O compensation cell disabled
1: I/O compensation cell enabled

14.5.13 SBS compensation cell for I/Os value register (SBS_CCVALR)

Address offset: 0x114

Reset value: 0x0000 0088

Reset: system reset

Register security: depends on SBSSEC in SBS_SECCFGR

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APSRC2[3:0]				ANSRC2[3:0]				APSRC1[3:0]				ANSRC1[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **APSRC2[3:0]**: compensation value for the PMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

Bits 11:8 **ANSRC2[3:0]**: Compensation value for the NMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

Bits 7:4 **APSRC1[3:0]**: compensation value for the PMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

Bits 3:0 **ANSRC1[3:0]**: compensation value for the NMOS transistor
This value is provided by the cell and must be interpreted by the processor to compensate the slew rate in the functional range.

14.5.14 SBS compensation cell for I/Os software code register (SBS_CCSWCR)

Address offset: 0x118

Reset value: 0x0000 7878

Reset: system reset

Register security: depends on SBSSEC in SBS_SECCFGR

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW_APSRC2[3:0]				SW_ANSRC2[3:0]				SW_APSRC1[3:0]				SW_ANSRC1[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **SW_APSRC2[3:0]**: PMOS compensation code for the V_{DDIO} power rails
This bitfield is written by software to define an I/O compensation cell code for PMOS transistors of the VDDIO power rail. This code is applied to the I/O when CS2 is set in SBS_CCSR.

Bits 11:8 **SW_ANSRC2[3:0]**: NMOS compensation code for VDDIO power rails
This bitfield is written by software to define an I/O compensation cell code for NMOS transistors of the VDD power rail. This code is applied to the I/O when CS2 is set in SBS_CCSR.

Bits 7:4 **SW_APSRC1[3:0]**: PMOS compensation code for the VDD power rails
This bitfield is written by software to define an I/O compensation cell code for PMOS transistors of the VDDIO power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.

Bits 3:0 **SW_ANSRC1[3:0]**: NMOS compensation code for VDD power rails
This bitfield is written by software to define an I/O compensation cell code for NMOS transistors of the VDD power rail. This code is applied to the I/O when CS1 is set in SBS_CCSR.

14.5.15 SBS Class B register (SBS_CFGR2)

Address offset: 0x120

Reset value: 0x0000 0000

Reset: system reset

Register security: depends on CLASSBSEC in SBS_SECCFGR

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SEL	CLL
												rs	rs	rs	rs

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

Bit 3 **ECCL**: ECC lock

This bit is set and cleared by software. It can be used to enable and lock the flash memory double ECC error with break input of TIM1/8/15/6/17.

0: double ECC error flag disconnected to timer break inputs

1: double ECC error flag connected to timer break inputs

Bit 2 **PVDL**: PVD lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection with TIM1/8/15/16/17 break inputs.

0: PVD interrupt disconnected from timer break inputs. PVD_EN and PVD_SEL[2:0] in the PWR registers are read/write.

1: PVD interrupt is connected to timer break inputs. PVD_EN and PVD_SEL[2:0] in the PWR registers are read only

Bit 1 **SEL**: SRAM ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the SRAM double ECC error signal with break input of TIM1/8/15/16/17.

0: SRAM double ECC error flag disconnected from timer break inputs

1: SRAM double ECC error flag connected to timer break inputs

Bit 0 **CLL**: core lockup lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the lockup (HardFault) output of Cortex-M33 with TIM1/8/15/16/17 break inputs.

0: lockup output disconnected from timer break inputs

1: lockup output connected to timer break inputs

14.5.16 SBS CPU nonsecure lock register (SBS_CNSLCKR)

Address offset: 0x144

Reset value: 0x0000 0000

Reset: system reset

Register security: This register can be read and written by privileged access only. Unprivileged access is RAZ/WI.

This register is used to lock the configuration of nonsecure MPU and VTOR_NS registers of the Cortex-M33.

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

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

Bit 1 **LOCKNSMPU**: nonsecure MPU register lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to nonsecure MPU_CTRL_NS, MPU_RNR_NS and MPU_RBAR_NS registers.

0: nonsecure MPU registers write enabled

1: nonsecure MPU registers write disabled

Bit 0 **LOCKNSVTOR**: VTOR_NS register lock

This bit is set by software and cleared only by a system reset.

0: VTOR_NS register write enabled

1: VTOR_NS register write disabled

14.5.17 SBS CPU secure lock register (SBS_CSLCKR)

Address offset: 0x148

Reset value: 0x0000 0000

Reset: system reset

Register security: This register can be written only when the access is secure/privilege.

A nonsecure read/write access is RAZ/WI and generates an illegal access event. When the system is not secure (TZ_STATE = 0), this register is RAZ/WI.

This register is used to lock the configuration of PRIS and BFHFNMINS in the AIRCR, SAU, secure MPU and VTOR_S registers of the Cortex-M33.

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

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

Bit 2 **LOCKSAU**: SAU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to SAU_CTRL, SAU_RNR, SAU_RBAR and SAU_RLAR registers.

0: SAU registers write enabled

1: SAU registers write disabled

Bit 1 **LOCKSMPU**: secure MPU registers lock

This bit is set by software and cleared only by a system reset. When set, this bit disables write access to secure MPU_CTRL, MPU_RNR and MPU_RBAR registers.

0: Secure MPU registers writes enabled

1: Secure MPU registers writes disabled

Bit 0 **LOCKSVTAIRCR**: VTOR_S and AIRCR register lock
This bit is set by software and cleared only by a system reset. When set, this bit disables write access to VTOR_S register, PRIS and BFHFNMINS bits in the AIRCR register.
0: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write enabled
1: VTOR_S register PRIS and BFHFNMINS bits in the AIRCR register write disabled

14.5.18 SBS flift ECC NMI mask register (SBS_ECCNMIR)

Address offset: 0x14C

Reset value: 0x0000 0000

Reset: system reset

Register security: secure access only when TZ_STATE = 1 (RAZ/WI in nonsecure access). No security protection if TZ_STATE = 0.

This register is accessible only through privilege transaction.

This register sets up the expected behavior on NMI regarding double ECC errors from the flash memory.

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

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

Bit 0 **ECCNMI_MASK_EN**: NMI behavior setup when a double ECC error occurs on flitf data part
0: NMI generated if a double ECC error in the flitf data part
1: NMI not generated if a double ECC error in the flitf data part

14.5.19 SBS register map

Table 132. SBS register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	**SBS_HDPLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR_HDPL[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x014	**SBS_HDPLSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									x	x	x	x	x	x	x	x
0x018	**SBS_ NEXTHDPLCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NEXTHDPL[1:0]	
	Reset value																															0	0
0x020	**SBS_DBGCR**	DBG_AUTH_SEC[7:0]								DBG_AUTH_HDPL[7:0]								DBG_UNLOCK[7:0]								AP_UNLOCK[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	**SBS_DBGLOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGCFG_LOCK[7:0]							
	Reset value																									1	0	1	1	0	1	0	0
0x028 to 0x030	**Reserved**	Reserved																															
0x034	**SBS_RSSCMDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSSCMD[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x09C	**Reserved**	Reserved																															
0x0A0	**SBS_ EPOCHSELCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCJ_SEL[1:0]	
	Reset value																															0	0
0x0A4 to 0x0BC	**Reserved**	Reserved																															
0x0C0	**SBS_SECCFGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPUSEC	Res.	CLASSBSEC	SBSSEC
	Reset value																													0		0	0
0x0C4 to 0x0FC	**Reserved**	Reserved																															
0x100	**SBS_PMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH_SEL_PHY [2:0](1)			Res.	PB9_FMP	PB8_FMP	PB7_FMP	PB6_FMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0		0	0	0	0																
0x104	**SBS_FPUIMR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IE[5:0]					
	Reset value																											1	1	1	1	1	1
0x108	**SBS_MESR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPMEE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCLR
	Reset value																0																X
0x110	**SBS_CCCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDY2	RDY1	Res.	Res.	Res.	Res.	CS2	EN2	CS1	EN1
	Reset value																							0	0					0	0	0	0

Table 132. SBS register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x114	**SBS_CCVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APSRC2 [3:0]				ANSRC2 [3:0]				APSRC1 [3:0]				ANSRC1 [3:0]			
	Reset value																	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
0x118	**SBS_CCSWCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW_APSRC2 [3:0]				SW_ANSRC2 [3:0]				SW_APSRC1 [3:0]				SW_ANSRC1 [3:0]			
	Reset value																	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0
0x120	**SBS_CFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCL	PVDL	SEL	CLL
	Reset value																													0	0	0	0
0x124 to 0x140	**Reserved**	Reserved																															
0x144	**SBS_CNSLCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKNSMPU	LOCKNSVTOR
	Reset value																															0	0
0x148	**SBS_CSLCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKSAU	LOCKSMPU	LOCKSVTAIRCR
	Reset value																														0	0	0
0x14C	**SBS_ECCNMIR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCNMI_MASK_EN
	Reset value																																0

1. Refer to the product datasheet for the availability of Ethernet. If not present, consider the associated bits as reserved, and keep them at reset value.

Refer to *Section 2.3* for the register boundary addresses.

15 Peripherals interconnect matrix

15.1 Interconnect matrix introduction

Several peripherals have direct connections between them, enabling autonomous communication and/or synchronization: this approach saves CPU resources, and power supply consumption. In addition, these hardware connections remove software latency and help the design of predictable system.

Depending on peripherals, these interconnections can operate in Run, Sleep, and Stop modes.

15.2 Connection summary

Table 133. Peripherals interconnect matrix(1)(2)(3)

Source	Destination																				
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM12	TIM15	TIM16	TIM17	LPTIM1/2/3	LPTIM4	LPTIM5/6	ADC1/2	DAC1/2	GPDMA1/2	TAMP	RTC	AES/SAES
TIM1	-	*1*	*1*	*1*	*1*	*1*	-	-	*1*	*1*	-	-	-	-	-	*2*	*4*	-	-	-	-
TIM8	*1*	-	*1*	*1*	*1*	*1*	-	-	*1*	*1*	-	-	-	-	-	*2*	*4*	-	-	-	-
TIM2	*1*	*1*	-	*1*	*1*	*1*	-	-	*1*	*1*	-	-	-	-	-	*2*	*4*	*10*	-	-	-
TIM3	*1*	*1*	*1*	-	*1*	*1*	-	-	*1*	*1*	-	-	-	-	-	*2*	-	-	-	-	-
TIM4	*1*	*1*	*1*	*1*	-	*1*	-	-	*1*	*1*	-	-	-	-	-	*2*	*4*	-	-	-	-
TIM5	*1*	*1*	*1*	*1*	*1*	-	-	-	*1*	*1*	-	-	-	-	-	-	*4*	-	-	-	-
TIM6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*2*	*4*	-	-	-	-
TIM7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*4*	-	-	-	-
TIM12	*1*	*1*	*1*	*1*	*1*	*1*	-	-	-	-	-	-	-	-	-	-	*4*	*10*	-	-	-
TIM13	*1*	*1*	*1*	*1*	*1*	*1*	-	-	*1*	*1*	-	-	-	-	-	-	-	-	-	-	-
TIM14	*1*	*1*	*1*	*1*	*1*	*1*	-	-	*1*	*1*	-	-	-	-	-	-	-	-	-	-	-
TIM15	*1*	*1*	*1*	*1*	*1*	-	-	-	*1*	-	-	-	-	-		*2*	*4*	*10*	-	-	-
TIM16	*1*	*1*	*1*	*1*	*1*	-	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-
TIM17	*1*	*1*	*1*	*1*	*1*	-	-	-	*1*	-	-	-	-	-	-	-	-	-	-	-	-
LPTIM1/2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*2*	*4*	*10*	-	-	-
LPTIM3/4/5/6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*10*	-	-	-
ADC1	*3*	*3*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
ADC2	*3*	*3*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-
GPDMA1/2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*10*	-	-	-
EXTI	-	-	-	-	-	-	-	-	-	-	-	-	*6*	*6*	*6*	*2*	*4*	*10*	-	-	-
RTC wake-up	-	-	-	-	-	*7*	-	-	-	-	*7*	-	-	-	-	-	-	*10*	-	-	-
RTC alarm	-	-	-	-	-	-	-	-	-	-	-	-	*6*	*6*	*6*	-	-	*10*	-	-	-
TAMP	-	-	-	-	-	-	-	-	-	-	-	-	*6*	*6*	*6*	-	-	*10*	-	*14*	*15*
HSE	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	-	-	-
LSE	-	-	*5*	-	-	-	-	-	-	*5*	-	*5*	-	-	-	-	*12*	-	-	-	-
CSS in LSE	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-
CSI	-	-	-	-	-	-	-	-	*5*	*5*	-	-	-	-	-	-	-	-	-	-	
HSI	-	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	-	-	-	-	-	-
LSI	-	-	-	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	-	-	-	-

Table 133. Peripherals interconnect matrix(1)(2)(3) (continued)

Source	Destination																				
	TIM1	TIM8	TIM2	TIM3	TIM4	TIM5	TIM6	TIM7	TIM12	TIM15	TIM16	TIM17	LPTIM1/2/3	LPTIM4	LPTIM5/6	ADC1/2	DAC1/2	GPDMA1/2	TAMP	RTC	AES/SAES
MCO1	-	-	-	-	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	-	-	-
MCO2	-	-	-	-	-	-	-	-	-	*5*	-	-	-	-	-	-	-	-	-	-	-
V_{CORE}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	-	-	-	-	-
V_{REFINT}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	-	-	-	-	-
V_{sensor}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	-	-	-	-	-
V_{BAT8}	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*11*	-	-	-	-	-
V_{BAT} / Temp. monitor	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*13*	-	-
System errors	*8*	*8*	-	-	-	-	-	-	-	*8*	*8*	*8*	-	-	-	-	-	-	-	-	-
System flash memory	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*15*
AES/SAES	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*15*
Ethernet	-	-	*9*	*9*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

1. Numbers in this table are links to corresponding subsections of *Section 15.3*.
2. "-" means no interconnect.
3. .Refer to the product datasheet for the availability of peripherals. If the peripheral is not present, consider the associated interconnection as unavailable.

15.3 Interconnection details

15.3.1 Master to slave interconnection for timers

From timer (TIM1/2/3/4/5/8/12/13/14/15/16/17) to timer (TIM1/2/3/4/5/8/12/15).

Purpose

Some timers are linked together internally for synchronization or chaining.

When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer configured in slave mode.

A description of the feature is provided in *Section 39.4.23: Timer synchronization*.

The synchronization modes are detailed in:

- *Section 38.3.30* for advanced-control timers TIM1/8
- *Section 39.4.22* for general-purpose timers TIM2/3/4/5
- *Section 41* for the general-purpose timers TIM12/13/14
- *Section 42.4.23* for the general-purpose timer TIM15

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1/8) following a configurable timer event. It can be also from signals tim16_oc1 and tim17_oc1 in case of TIM16/17. The input (to slave) is on signals TIMx_ITR0/1/2/3.

The possible master/slave connections are given in:

- *Table 398* for advanced-control timers TIM1/8
- *Table 421* for general-purpose timers TIM2/3/4/5
- *Table 442* for the general-purpose timers TIM12/13/14
- *Table 454* for the general-purpose timer TIM15

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 410: Effect of low-power modes on TIM1/TIM8*
- *Table 428: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 444: Effect of low-power modes on TIM12/TIM13/TIM14*
- *Table 460: Effect of low-power modes on TIM15/TIM16/TIM17*

15.3.2 Triggers to ADCs

From EXTI, timers (TIM1/2/3/4/6/8/15) and LP timers (LPTIM1/2) to ADC1/ADC2.

Purpose

A conversion (or a sequence of conversions) can be triggered either by software or by an external event (such as timer capture or input pins). For ADC12, if the EXTEN[1:0] (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, external events can trigger a conversion with the selected polarity.

More details in:

- *Section 26.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*
- EXTEN[1:0] defined in *ADC configuration register (ADC_CFGR)*
- JEXTEN[1:0] defined in *ADC injected sequence register (ADC_JSQR)*

General-purpose timers (TIM2/3/4), basic timer (TIM6), advanced-control timers (TIM1/8), and general-purpose timer (TIM15) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo.

Low-power timers (LPTIM1/2) can be used to generate the ADC triggering event through the LPTIM channels, in addition to the EXTI on channels 11 and 15.

Triggering signals

For ADC1/ADC2, the input triggering signals and the description of the interconnection between ADC1/ADC2 and timers are given in:

- adc_ext_trg: *Table 257: ADC interconnection*
- adc_jext_trg: *Table 257: ADC interconnection*
- *Section 26.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*
- *Section 26.4.25: Timing diagrams example (single/continuous modes, hardware/software triggers)*

Active power mode

This interconnection is active in Run and Sleep modes for all ADCs. The timers are active only in Run and Sleep modes. The effects of low-power modes are given in:

- *Table 410: Effect of low-power modes on TIM1/TIM8*
- *Table 428: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 460: Effect of low-power modes on TIM15/TIM16/TIM17*
- *Table 466: STM32H5 LPTIM features*
- *Table 481: Effect of low-power modes on the LPTIM*

15.3.3 ADC analog watchdogs as triggers to timers

From ADC1/ADC2 to TIM1/8.

Purpose

The internal analog watchdog output signals coming from ADC1/ADC2 are connected to on-chip timers. ADC1/ADC2 can provide trigger events through analog watchdog signals to advanced-control timers (TIM1/8) to reset, start, stop, or clock the counter.

Settings description of the ADC analog watchdog and timer trigger, are provided in:

- *Section 38.3.6: External trigger input* for TIM1/8
- *Table 399* for the internal ADC1/ADC2 sources connected to TIM1/8 (tim_etr) input multiplexer
- *Section 26.4.28* for the ADC1/ADC2/ADC_AWDy_OUT signal output generation

Triggering signals

The output (from ADC) is on signals ADCn_AWDx_OUT, with n being the ADC instance and x = 1, 2, 3 (three watchdogs per ADC). The input (to timer) is on signal TIMx_ETR (external trigger).

Active power mode

ADC1/ADC2 are active in Run and Sleep modes.

15.3.4 Triggers to DAC

From timer (TIM1/2/4/5/6/7/8/15), low-power timers (LPTIM1/2), and EXTI to DAC.

Purpose

General-purpose timers (TIM2/4/5/15), basic timers (TIM6/7), advanced-control timers (TIM1/8), LP timers (LPTIM1/2) outputs channels (lptim1_ch1 and lptim2_ch1), and EXTI can be used as triggering event to start a DAC conversion.

Triggering signals

The output (from timer) on the TIMx_TRGO signal and from LP timers are directly connected to corresponding DAC inputs.

The selection of input triggers on DAC is provided in:

- *Table 281: DAC interconnection*
- *Section 28.4.8: DAC trigger selection*

Active power mode

This interconnect is active in Run, Sleep, and Stop modes.

15.3.5 Clock sources to timers

From HSE, LSE, LSI, HSI, and MCO to timers (TIM2/12/15/16/17) and LP timers (LPTIM1/2).

Purpose

A timer input or counter can receive different clock sources, and can be used, for example, to calibrate the internal oscillator on a reference clock.

External clocks (HSE, LSE), internal clocks (LSI, CSI, HSI), and microcontroller output clock (MCO) can be used as input to timers:

- LSE, HSI, and CSI are assigned to general purpose timers TIM2/12 as external inputs signals. LSE can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in). Inputs assignment and clock selection description are detailed in:
  - *Section 39.4.5: Clock selection* for TIM2
  - External clock mode1: *Table 440: Interconnect to the tim_ti1 input multiplexer* for TIM12, tim_ti1_in4 (HSI), and tim_ti1_in5 (CSI)
  - External clock mode2: *Table 422: Interconnect to the tim_etr input multiplexer* for tim_etr3 (LSE)
- LSE, LSI, CSI, and HSE are assigned to general purpose timers TIM15/16/17 as external inputs signals. LSE/LSI/CSI/HSE can be selected as counter clock provided by an external clock source in mode1 (tim_ti1 or tim_ti2 signals). Inputs assignment and clock selection description are detailed in:
  - *Section 42.4.6: Clock selection* for TIM15/16/17. External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
  - *Table 440: Interconnect to the tim_ti1 input multiplexer*, tim_ti1_in1 (LSI-TIM16), tim_ti1_in2 (LSE-TIM16/HSE-TIM17), tim_ti1_in4 (LSE- TIM15), and tim_ti1_in5 (CSI-TIM15)
- Microcontroller output clock (MCO): MCO1 is connected as external input to general-purpose timer TIM17, MCO2 is connected as external input to general-purpose timer TIM15.
  - *Table 452: Interconnect to the tim_ti1 input multiplexer* for TIM15/TIM16/TIM17
- LSI and LSE can be selected as input capture 2 to LPTIM1 as described in *Table 475: LPTIM1 input capture 2 connections*.
- HSI/1024, CSI/128, and HSI/8 can be selected as input capture 2 to LPTIM2 as described in *Table 476: LPTIM2 input capture 2 connections*.

Triggering signals

lptim_ic2_mux1 LPTIM input capture selection can be set in the LPTIM configuration register 2 (LPTIM_CFGR2). For timers, the internal clock signal can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in).

Active power mode

This feature is available under Run and Sleep modes.

15.3.6 Triggers to low-power timers

From EXTI, TAMP, and RTC alarm to LP timers (LPTIM1/2/3/4/5/6).

Purpose

LPTIM1/2/3/4/5/6 counters can be started either by software, or after the detection of an active edge on one of the eight trigger inputs (see *Section 43.4.7: Trigger multiplexer*).

GPIO can also be selected as LPTIM input capture selection or LPTIM input selection, according to the LPTIM configuration register 2 (LPTIM_CFGR2).

Triggering signals

This trigger feature is described in *Section 43.4.7: Trigger multiplexer* and the following sections. The input selection is described in *Table 471: LPTIM1/2/3/4/5/6 external trigger connections*.

Active power mode

This interconnection is active in Run, Sleep, and Stop modes.

15.3.7 RTC wake-up as inputs to timers

From RTC to timer (TIM16).

Purpose

RTC wake-up interrupt can be used as input to general-purpose timer (TIM16) channel 1.

Triggering signals

RTC wake-up signal is connected to tim_ti1_in3 signal as described in *Table 452: Interconnect to the tim_ti1 input multiplexer* for TIM16.

Active power mode

This interconnection is active down to Stop mode. Timers are not active but the count is performed at wake-up.

15.3.8 System errors as break signals to timers

From system errors to timers (TIM1/8/15/16/17).

Purpose

CSS, CPU lockup, SRAM2/3 ECC double errors, SRAM1 parity errors, FLASH ECC double-error detection, and PVD can generate system errors in the form of timer break toward timers (TIM1/8/15/16/17).

The purpose of the break function is to protect power switches driven by PWM signals generated by the timers.

Triggering signals

The possible sources of break are described in:

- *Section 38.3.18: Using the break function* for TIM1/8
- *Section 42.4.15: Using the break function* for TIM15/16/17
- *Table 402: System break interconnect* for TIM1/8
- *Table 456: System break interconnect* for TIM15/16/17

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:

- *Table 410: Effect of low-power modes on TIM1/TIM8*
- *Table 428: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 460: Effect of low-power modes on TIM15/TIM16/TIM17*

15.3.9 Triggers for communication peripherals

From Ethernet to timers (TIM2/TIM3).

Purpose

To synchronize system clock with network, internal connections are available between timers and PTP to check for clock drifts.

Triggering signals

- The outputs (from timer) are directly connected to Ethernet PTP triggers inputs. More details are given in the sections below:
- *Section 57.3: Ethernet pins and internal signals*
- *Table 417: Interconnect to the tim_ti1 input multiplexer*
- *Table 422: Interconnect to the tim_etr input multiplexer*

Active power mode

These interconnections remain active in Run and Sleep modes.

15.3.10 Triggers to GPDMA1/2

From EXTI, RTC (alarm/wake-up), TAMP, timers (TIM2/12/15), LP timers (LPTIM1/2/3/4/5/6), GPDMA1 transfer complete (gpdma1_chx_tc, gpdma2_chx_tcf) to GPDMA1/2.

Purpose

A GPDMA trigger can be assigned to GPDMA channel x. A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the LLI link transfer. More details are given in the sections below:

- *Section 16.3.7: GPDMA triggers*
- *Section 16.4.12: GPDMA triggered transfer*

- *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details on:
  - Trigger selection TRIGSEL[5:0] field
  - Trigger mode (LLI) defined by TRIGM[1:0]
  - Trigger polarity as defined by TRIGPOL[1:0]

Triggering signals

GPDMA trigger mapping is specified in *Table 140: Programmed GPDMA1/2 trigger*, according to GPDMA_CxTR2.TRIGSEL[5:0].

Active power mode

This interconnection remains functional in Sleep mode.

Refer to:

- *Section 16.6: GPDMA in low-power modes*

15.3.11 Internal analog signals to analog peripherals

From internal analog source to ADC (ADC1/2).

Purpose

The internal reference voltage (V_{REFINT}), the internal temperature sensor (V_{SENSE}), the internal digital core voltage (V_{DDCORE}), and the V_{BAT} monitoring channel are connected to ADC (ADC1/2) input channels.

This is according to:

- *Section 26.2: ADC main features*
- *Section 26.4.11: Channel selection (ADC_SQRy, ADC_JSQR)*

15.3.12 Clock source for the DAC sample and hold mode

From LSI/LSE to DAC1.

Purpose

DAC1 can run in Stop mode. The sample and hold block and its associated registers use the LSI or LSE clock source (dac_hold_ck) in Stop mode.

Table 280: DAC internal input/output signals: dac_hold_ck, Input, DAC low-power clock used in sample and hold mode

Active power mode

This feature remains available in Run, Sleep and Stop modes.

15.3.13 Internal tamper sources

From internal peripherals, clocks, or monitoring, to tamper.

Purpose

To detect any abnormal activity or tentative to corrupt the device, embedded tampers alert the system of undesired events. Different actions can be taken as consequence.

The list of tamper sources can be found in *Table 507: TAMP interconnection*.

Active power mode

This interconnection is active in all power modes if the tamper source is activated.

15.3.14 Output from tamper to RTC

From TAMP to RTC.

Purpose

The RTC can timestamp a tamper event to retrieve history of the detection. The RTC can also control GPIOs, and set a signal based on tamp or alarm status outside the MCU.

Refer to *Section 46.3.3: GPIOs controlled by the RTC and TAMP* for more details.

Active power mode

This interconnection remains active in all power modes.

15.3.15 Encryption keys to AES/SAES

From TAMP backup registers, system flash memory to and in between SAES and AES.

Purpose

The encryption mechanism requires an hardware key that must be stored in a protected non-volatile memory. Different approaches are implemented to load them in a non-readable way. Tamper backup registers or system flash memory can be used to store respectively BHK or RHUK, and to implement a dedicated bus to pass it to the SAES.

Refer to *Section 34.4.14: SAES operation with wrapped keys* for more details.

The AES encryption mechanism (faster than the SAES) can benefit from the sharing key of the SAES. Refer to *Section 34.4.15: SAES operation with shared keys* for more details.

Active power mode

AES and SAES are operating under Run and Sleep modes.

16 General purpose direct memory access controller (GPDMA)

16.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

16.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 8 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 16.3.2*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers

  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 16.3.2*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA half transfer and transfer complete events generation
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers
  - Channel abort and restart
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

16.3 GPDMA implementation

16.3.1 GPDMA instances

There are two instances of the GPDMA in the devices, named as GPDMA1 and GPDMA2.

Each GPDMA instance has the same channel-based implementation and is connected to the same requests and triggers, as detailed in the following sub-sections.

16.3.2 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 134. GPDMA1/2 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 3	2	0	Channel x (x = 0 to 3) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and SRAM.
x = 4 to 5	4	0	Channel x (x = 4 to 5) is implemented with: – a FIFO of 32 bytes, 8 words – fixed/contiguously incremented addressing These channels may be used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.
x = 6 to 7	4	1	Channel x (x = 6 to 7) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be used for GPDMA transfers between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.

16.3.3 GPDMA in low-power modes

The GPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 135. GPDMA1/2 wake-up in low-power modes

Feature	Low-power modes
Wake-up	GPDMA1/2 in Sleep mode

16.3.4 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[7:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 136. Programmed GPDMA1/2 request

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
0	adc1_dma
1	adc2_dma
2	dac1_ch1_dma
3	dac1_ch2_dma
4	tim6_upd_dma

Table 136. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
5	tim7_upd_dma
6	spi1_rx_dma
7	spi1_tx_dma
8	spi2_rx_dma
9	spi2_tx_dma
10	spi3_rx_dma
11	spi3_tx_dma
12	i2c1_rx_dma
13	i2c1_tx_dma
14	Reserved
15	i2c2_rx_dma
16	i2c2_tx_dma
17	Reserved
18	i2c3_rx_dma
19	i2c3_tx_dma
20	Reserved
21	usart1_rx_dma
22	usart1_tx_dma
23	usart2_rx_dma
24	usart2_tx_dma
25	usart3_rx_dma
26	usart3_tx_dma
27	uart4_rx_dma
28	uart4_tx_dma
29	uart5_rx_dma
30	uart5_tx_dma
31	usart6_rx_dma
32	usart6_tx_dma
33	uart7_rx_dma(1)
34	uart7_tx_dma(1)
35	uart8_rx_dma(1)
36	uart8_tx_dma(1)
37	uart9_rx_dma(1)
38	uart9_tx_dma(1)
39	uart10_rx_dma(1)

Table 136. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
40	uart10_tx_dma(1)
41	uart11_rx_dma(1)
42	uart11_tx_dma(1)
43	uart12_rx_dma(1)
44	uart12_tx_dma(1)
45	lpuart1_rx_dma
46	lpuart1_tx_dma
47	spi4_rx_dma
48	spi4_tx_dma
49	spi5_rx_dma(1)
50	spi5_tx_dma(1)
51	spi6_rx_dma(1)
52	spi6_tx_dma(1)
53	sai1_a_dma(1)
54	sai1_b_dma(1)
55	sai2_a_dma(1)
56	sai2_b_dma(1)
57	ospi1_dma
58	tim1_cc1_dma
59	tim1_cc2_dma
60	tim1_cc3_dma
61	tim1_cc4_dma
62	tim1_upd_dma
63	tim1_trg_dma
64	tim1_com_dma
65	tim8_cc1_dma
66	tim8_cc2_dma
67	tim8_cc3_dma
68	tim8_cc4_dma
69	tim8_upd_dma
70	tim8_tig_dma
71	tim8_com_dma
72	tim2_cc1_dma
73	tim2_cc2_dma
74	tim2_cc3_dma

Table 136. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
75	tim2_cc4_dma
76	tim2_upd_dma
77	tim3_cc1_dma
78	tim3_cc2_dma
79	tim3_cc3_dma
80	tim3_cc4_dma
81	tim3_upd_dma
82	tim3_trg_dma
83	tim4_cc1_dma
84	tim4_cc2_dma
85	tim4_cc3_dma
86	tim4_cc4_dma
87	tim4_upd_dma
88	tim5_cc1_dma
89	tim5_cc2_dma
90	tim5_cc3_dma
91	tim5_cc4_dma
92	tim5_upd_dma
93	tim5_trg_dma
94	tim15_cc1_dma
95	tim15_upd_dma
96	tim15_trg_dma
97	tim15_com_dma
98	tim16_cc1_dma(1)
99	tim16_upd_dma(1)
100	tim17_cc1_dma(1)
101	tim17_upd_dma(1)
102	lptim1_ic1_dma
103	lptim1_ic2_dma
104	lptim1_ue_dma
105	lptim2_ic1_dma
106	lptim2_ic2_dma
107	lptim2_ue_dma
108	dcmi_dma or pssi_dma(2)
109	aes_out_dma

Table 136. Programmed GPDMA1/2 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
110	aes_in_dma
111	hash_in_dma
112	ucpd1_rx_dma
113	ucpd1_tx_dma
114	cordic_read_dma(1)
115	cordic_write_dma(1)
116	fmac_read_dma(1)
117	fmac_write_dma(1)
118	saes_out_dma
119	saes_in_dma
120	i3c1_rx_dma
121	i3c1_tx_dma
122	i3c1_tc_dma
123	i3c1_rs_dma
124	i2c4_rx_dma(1)
125	i2c4_tx_dma(1)
126	Reserved
127	lptim3_ic1_dma(1)
128	lptim3_ic2_dma(1)
129	lptim3_ue_dma(1)
130	lptim5_ic1_dma(1)
131	lptim5_ic2_dma(1)
132	lptim5_ue_dma(1)
133	lptim6_ic1_dma(1)
134	lptim6_ic2_dma(1)
135	lptim6_ue_dma(1)
136	i3c2_rx_dma(3)
137	i3c2_tx_dma(3)
138	i3c2_tc_dma(3)
139	i3c2_rs_dma(3)

1. Not available in STM32H523/533 devices.
2. Depends on which exclusive function is used.
3. Not available in STM32H562/563/573 devices.

16.3.5 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 137. Programmed GPDMA1/2 request as a block request

GPDMA block requests
lptim1_ue_dma
lptim2_ue_dma
lptim3_ue_dma(1)
lptim5_ue_dma(1)
lptim6_ue_dma(1)

1. Not available in STM32H523/533 devices.

16.3.6 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 138. GPDMA1/2 channel with peripheral early termination

GPDMA channel x with peripheral early termination
x = 0 and x = 7

This GPDMA support is activated when the channel x is programmed with GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before that the GPDMA has transferred the whole block (see *Section 16.4.14* for more details).

Table 139. Programmed GPDMA1/2 request with peripheral early termination

Programmed GPDMA channel x request with peripheral early termination
i3c2_rx_dma
i2c2_rx_dma(1)

1. Not available in STM32H562/563/573 devices.

16.3.7 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[5:0] in GPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 140. Programmed GPDMA1/2 trigger

GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
0	exti0
1	exti1
2	exti2
3	exti3
4	exti4
5	exti5
6	exti6
7	exti7
8	tamp_trg1
9	tamp_trg2
10	tamp_trg3
11	lptim1_ch1
12	lptim1_ch2
13	lptim2_ch1
14	lptim2_ch2
15	rtc_alra_trg
16	rtc_alrb_trg
17	rtc_wut_trg
18	gpdma1_ch0_tc
19	gpdma1_ch1_tc
20	gpdma1_ch2_tc
21	gpdma1_ch3_tc
22	gpdma1_ch4_tc
23	gpdma1_ch5_tc
24	gpdma1_ch6_tc
25	gpdma1_ch7_tc
26	gpdma2_ch0_tc
27	gpdma2_ch1_tc
28	gpdma2_ch2_tc
29	gpdma2_ch3_tc
30	gpdma2_ch4_tc
31	gpdma2_ch5_tc
32	gpdma2_ch6_tc
33	gpdma2_ch7_tc
34	tim2_trgo

Table 140. Programmed GPDMA1/2 trigger (continued)

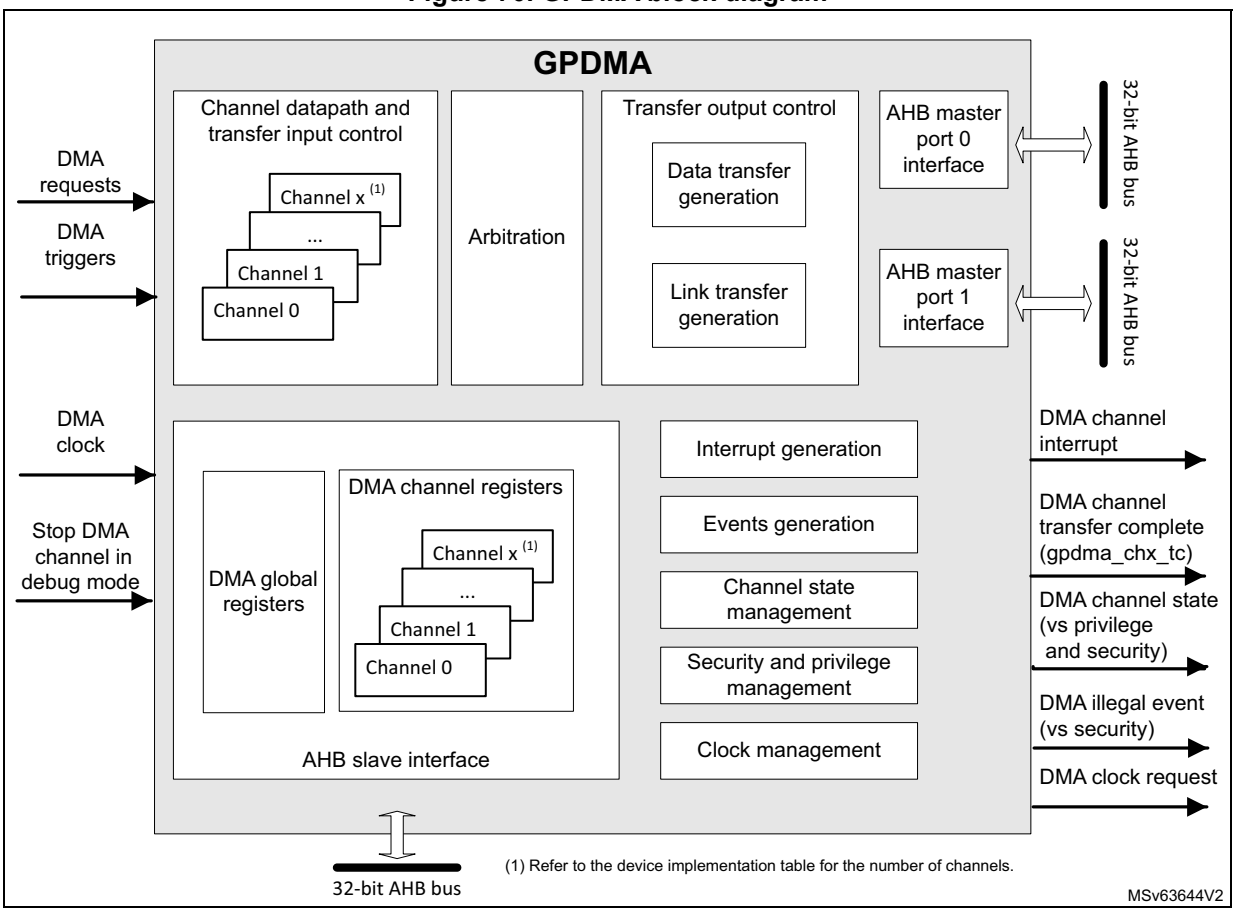
GPDMA_CxTR2.TRIGSEL[5:0]	Selected GPDMA trigger
35	tim15_trgo
36	tim12_trgo
37	lptim3_ch1(1)
38	lptim3_ch2(1)
39	lptim4_ait(1)
40	lptim5_ch1(1)
41	lptim5_ch2(1)
42	lptim6_ch1(1)
43	lptim6_ch2(1)
44	Reserved
45	EVENTOUT(2)

1. Not available in STM32H523/533 devices.
2. Not available in STM32H562/563/573 devices.

16.4 GPDMA functional description

16.4.1 GPDMA block diagram

Figure 70. GPDMA block diagram



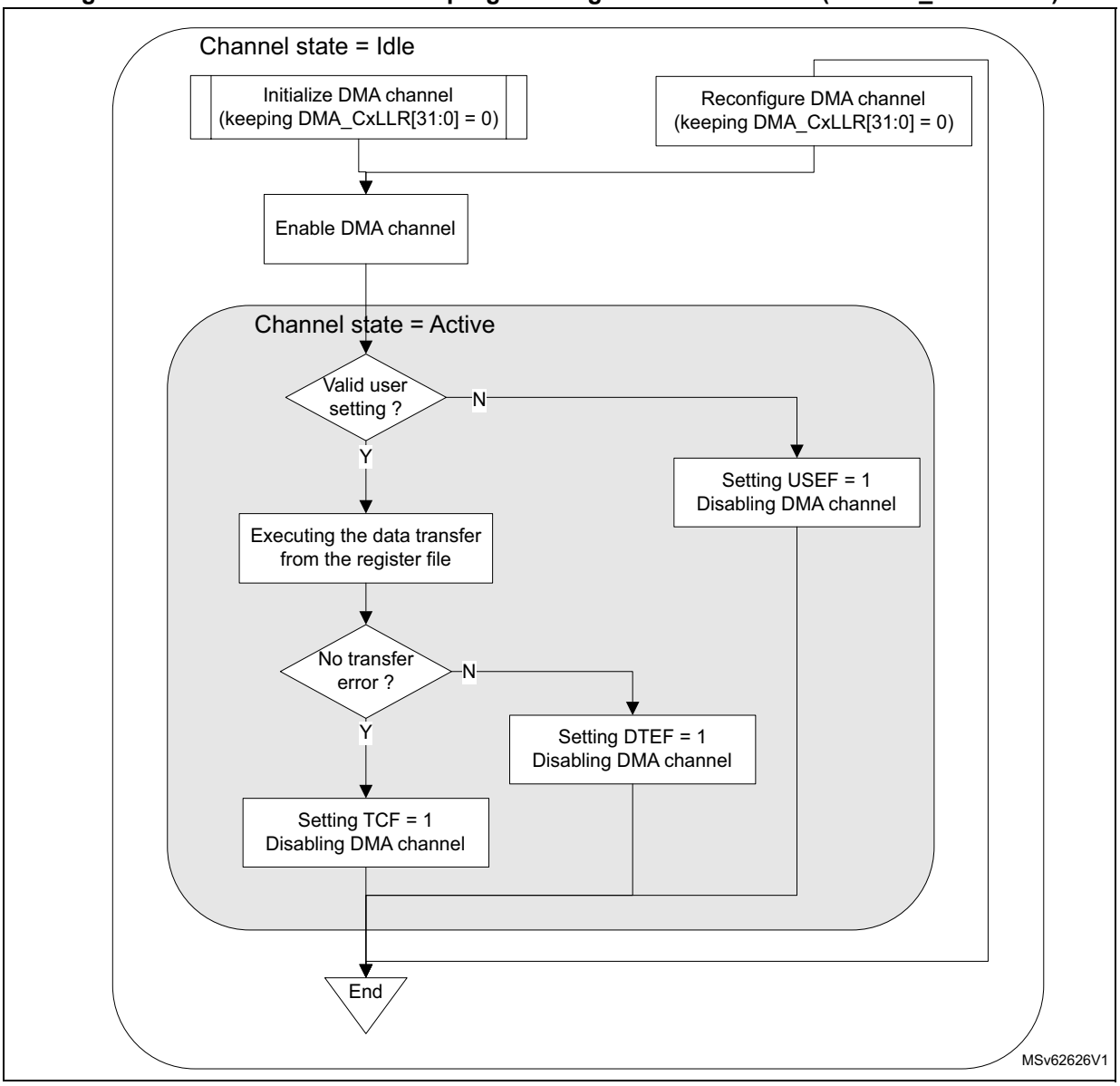
16.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

Figure 71 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 71. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



16.4.3 GPDMA channel suspend and resume

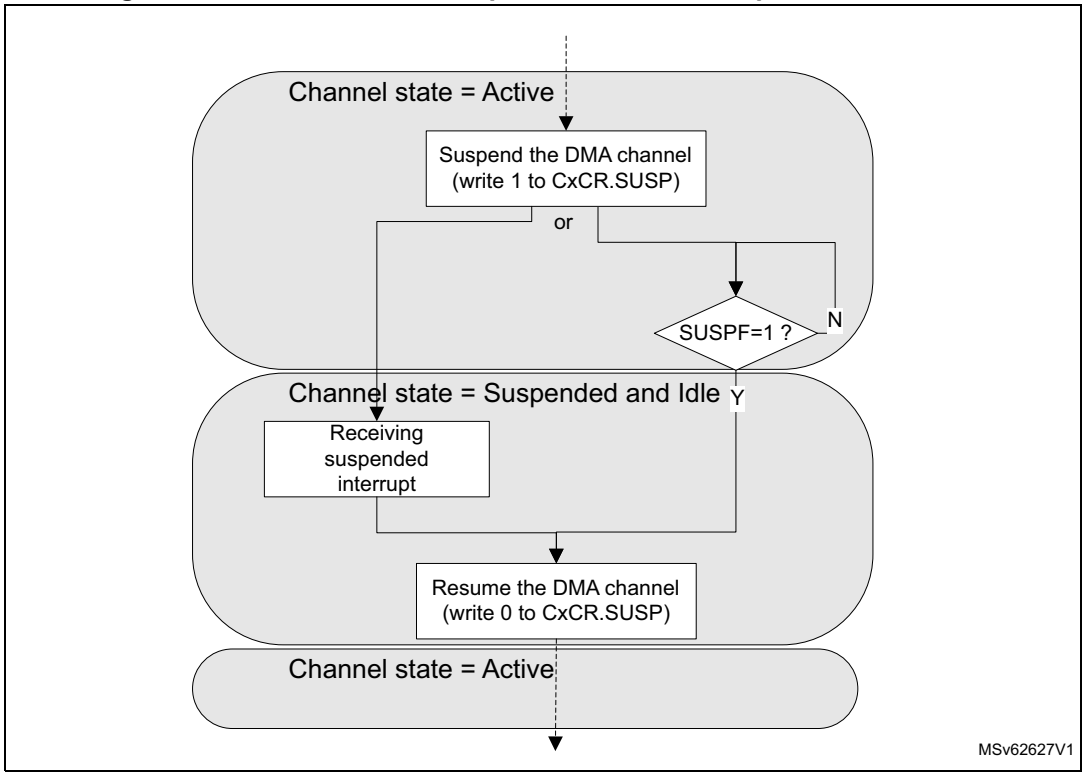
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note: An ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 72. GPDMA channel suspend and resume sequence



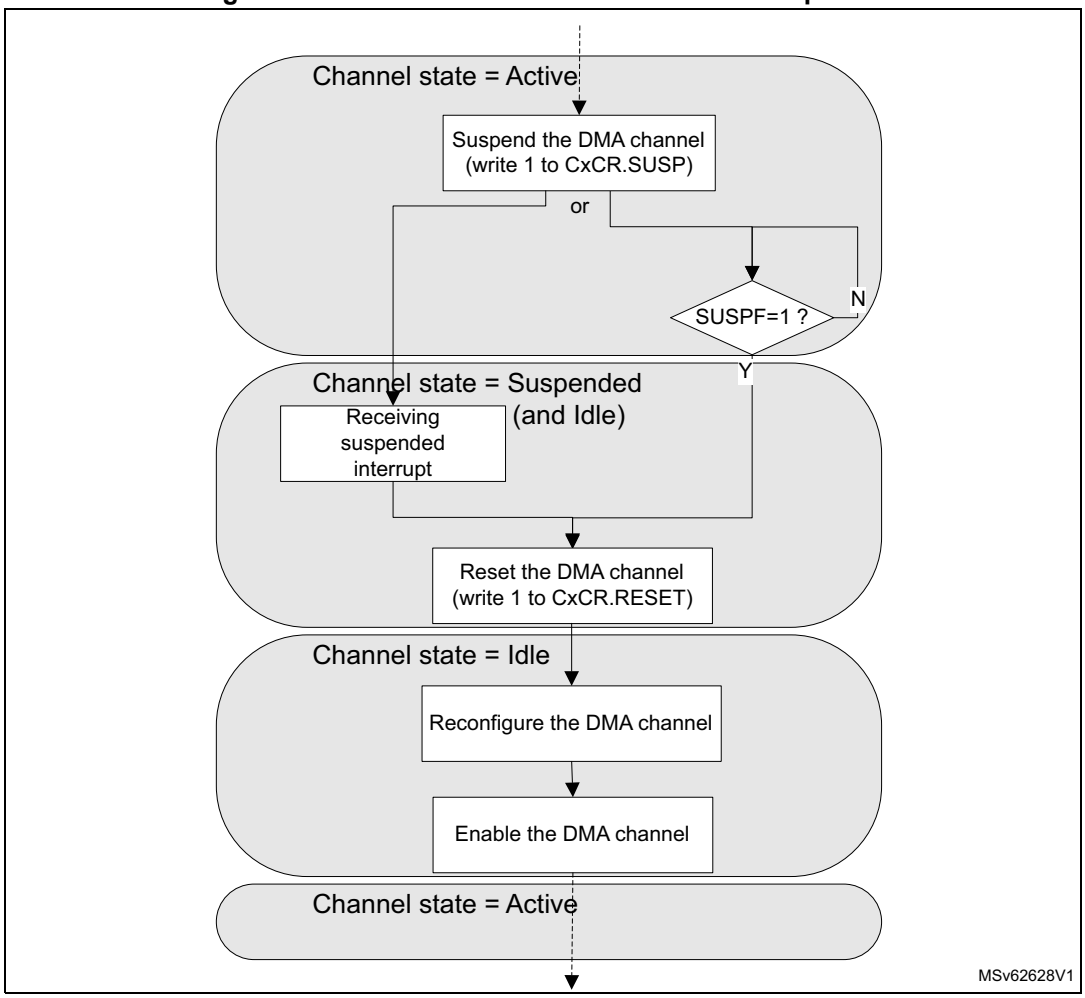
Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

16.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 73. GPDMA channel abort and restart sequence



16.4.5 GPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits, plus UB2 and UT3, in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3 or GPDMA_CxBR2) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the

GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 5) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 74*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory
- channel x (x = 6 to 7), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2 (see *Figure 75*), and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 74. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

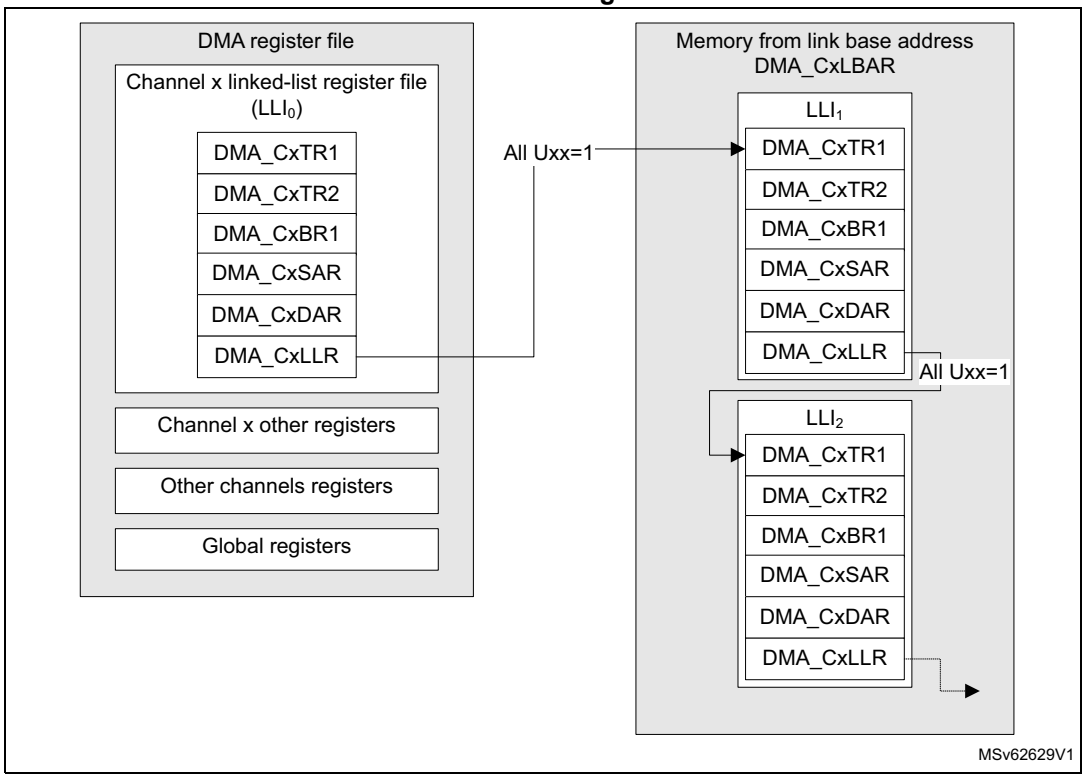
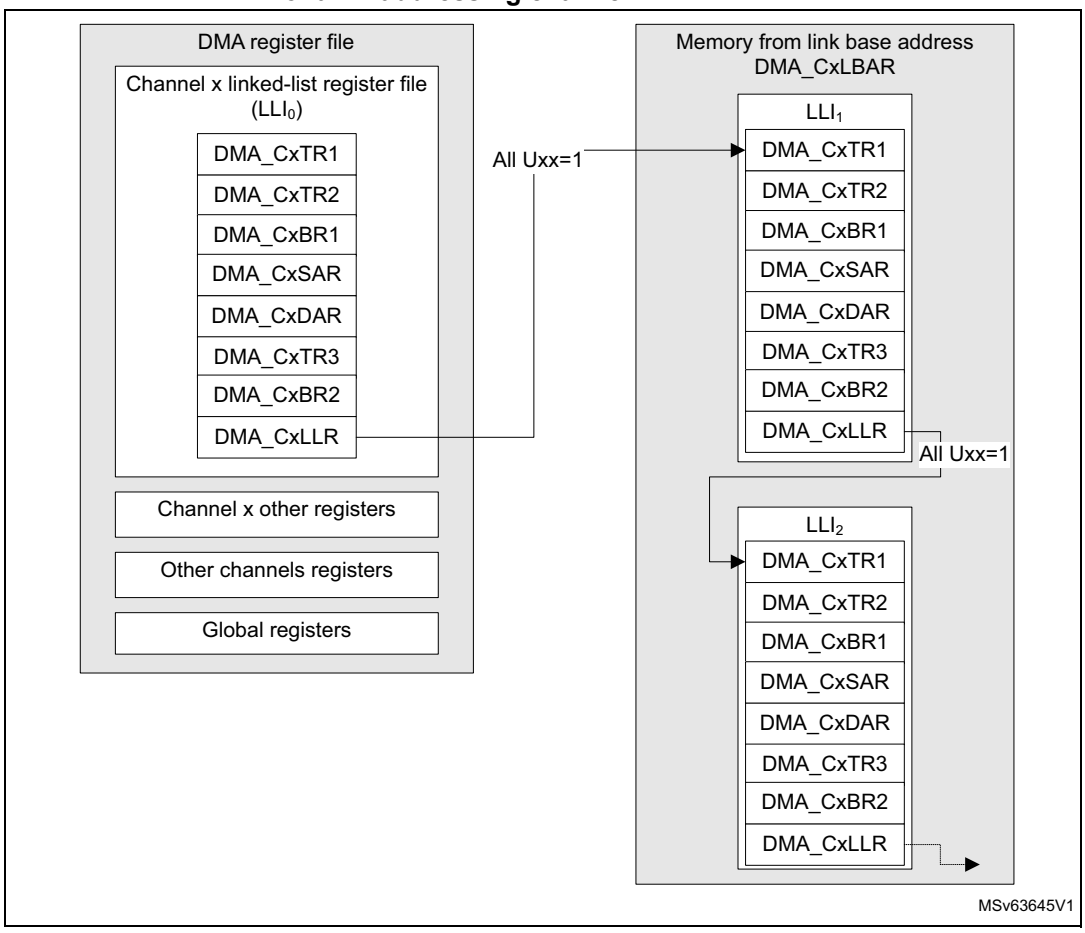


Figure 75. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 76*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 77*.

Figure 76. GPDMA dynamic linked-list data structure of a linear addressing channel x

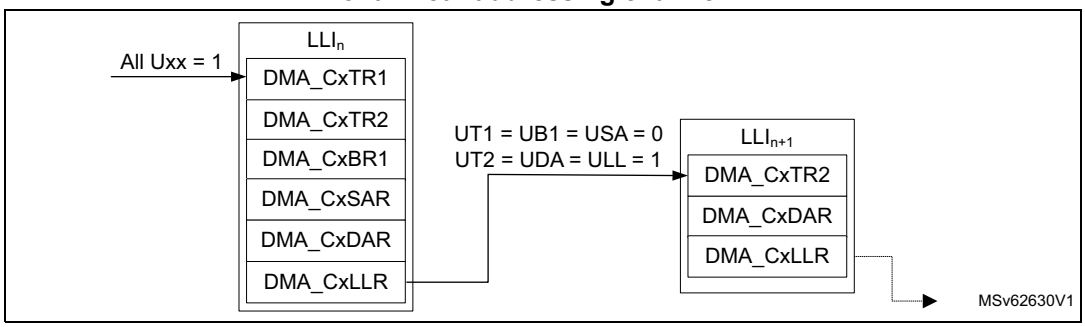
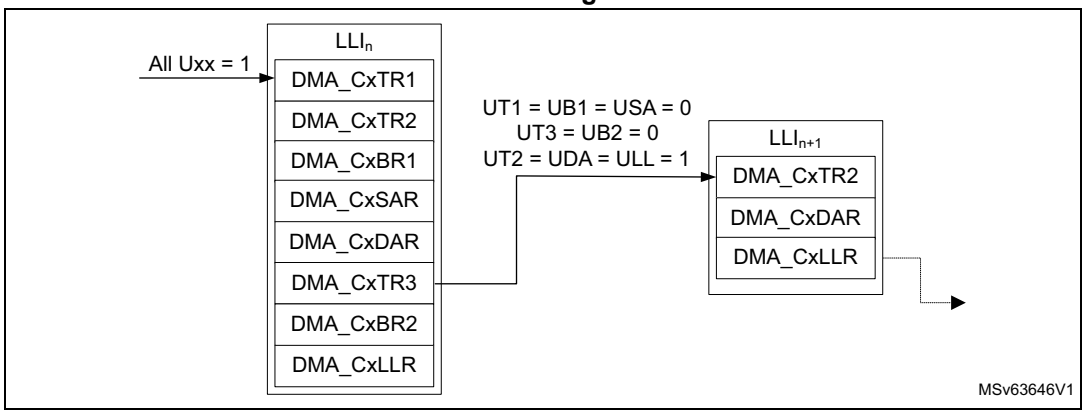


Figure 77. GPDMA dynamic linked-list data structure of a 2D addressing channel x



The user must program GPDMA_CxLLR for each LLI_n to be 32-bit aligned and not to exceed the 64-Kbyte addressable space pointed by GPDMA_CxLBAR.

16.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

16.4.7 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 16.4.14* for more details).

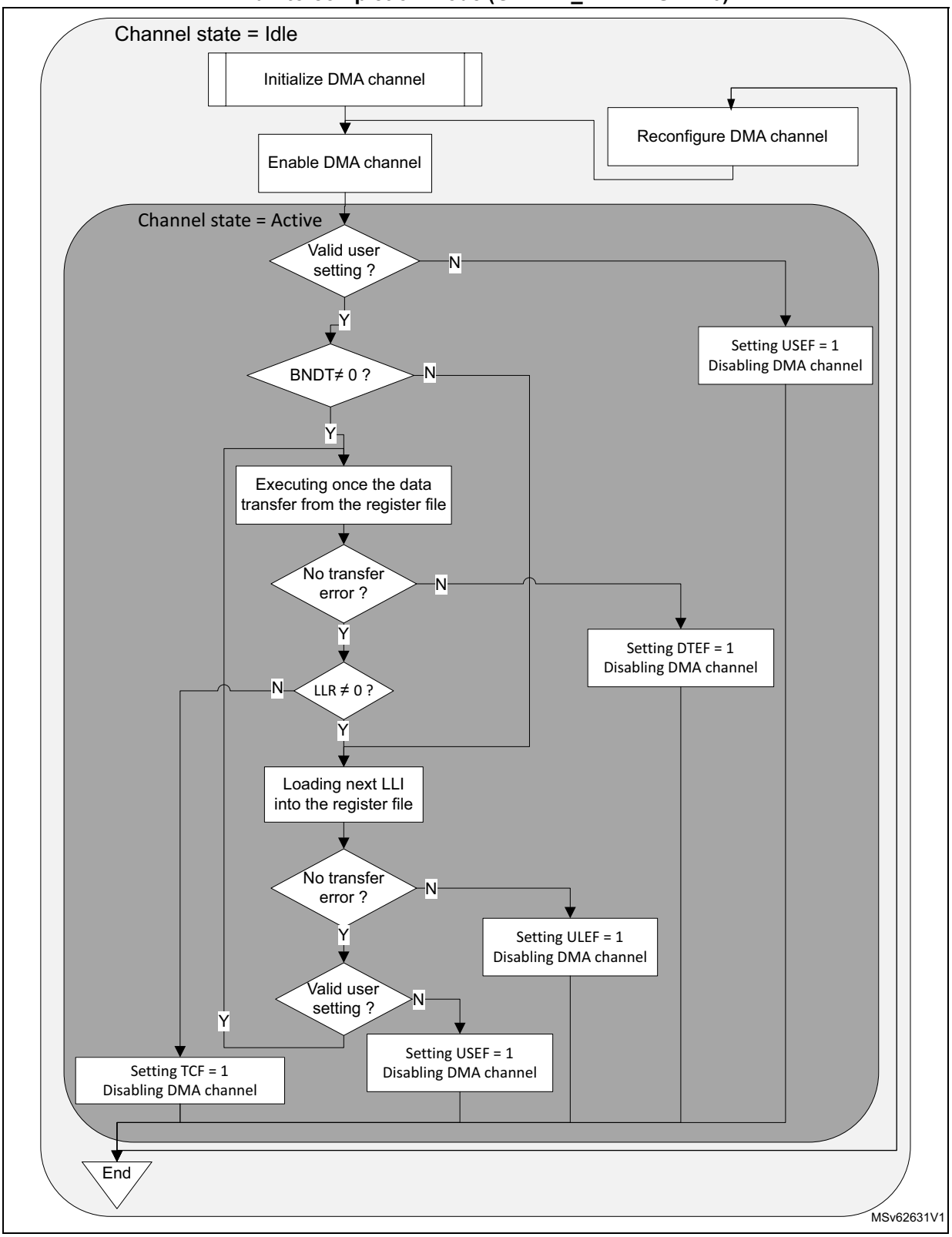
In the typical run-to-completion mode, the allocation of a GPDMA channel, including its fine programming, is done once during the GPDMA initialization. In order to have a reserved data communication link and GPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 78 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 78 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 78, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).

Figure 78. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

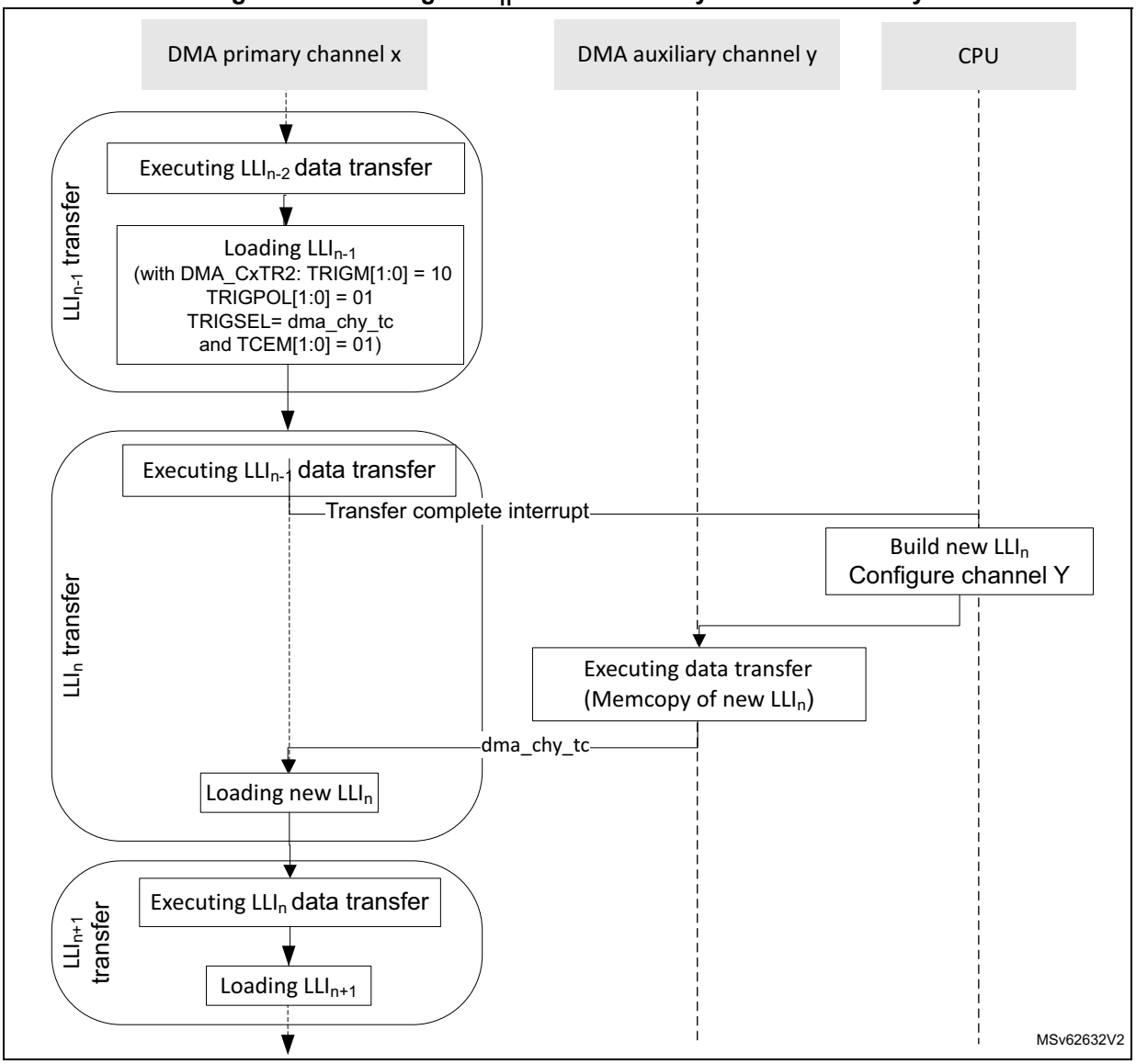
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see *Section 16.3.7* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 79 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 79. Inserting a LLI_n with an auxiliary GPDMA channel y



16.4.8 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 6 to 7), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 6 to 7) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 6 to 7) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 16.4.14* for more details).

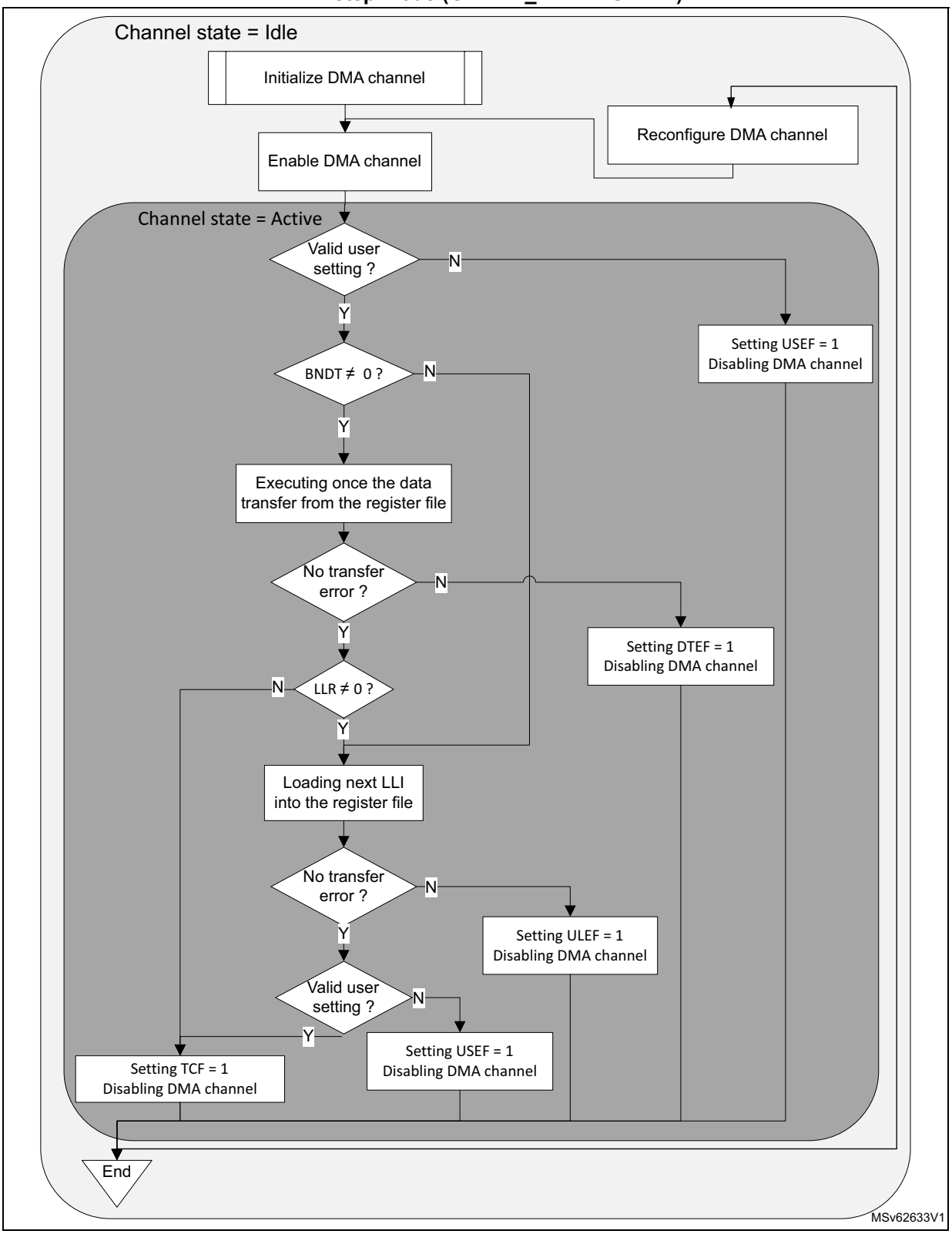
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 80 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 80 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 80, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).

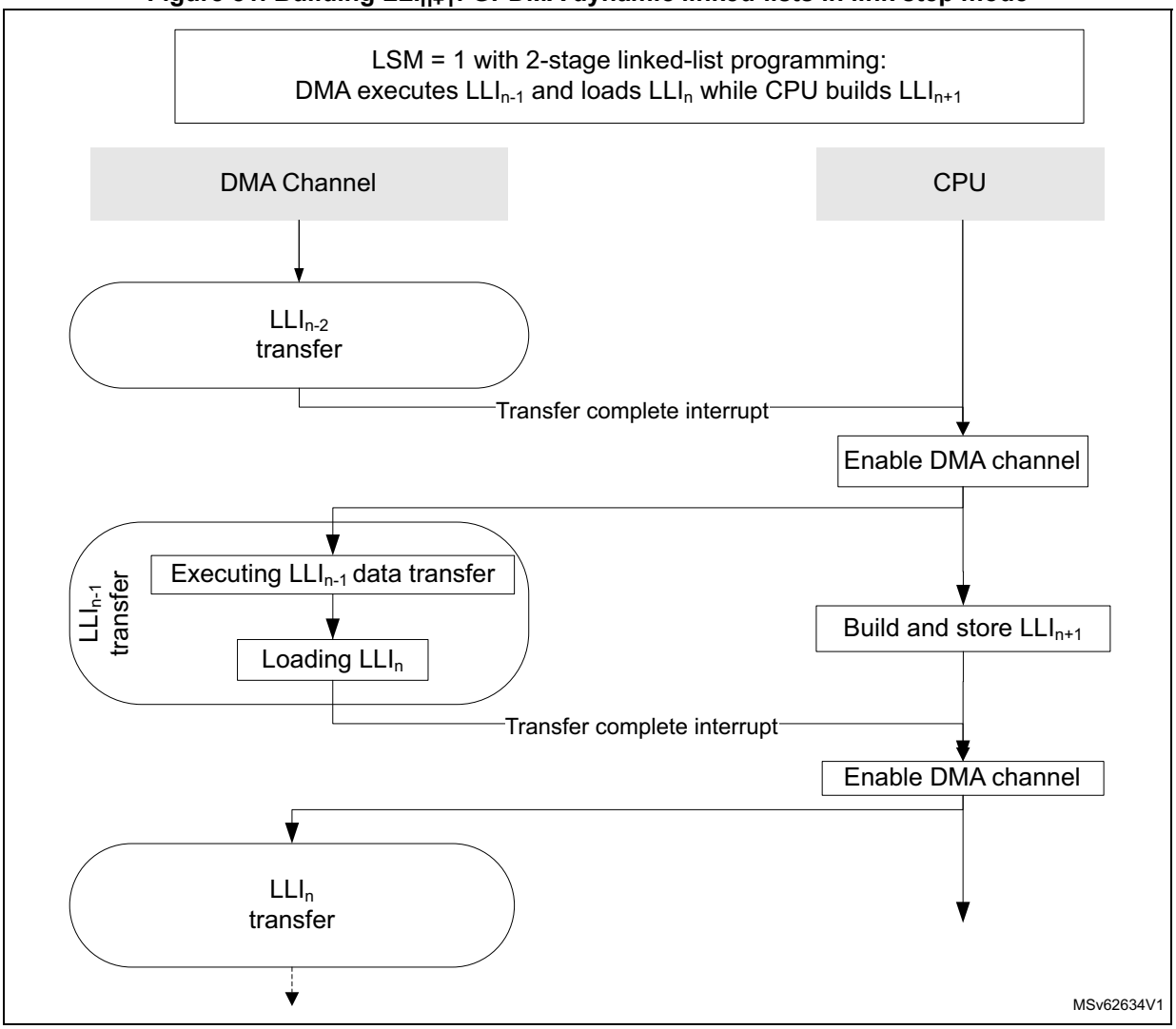
Figure 80. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in *Figure 81*.

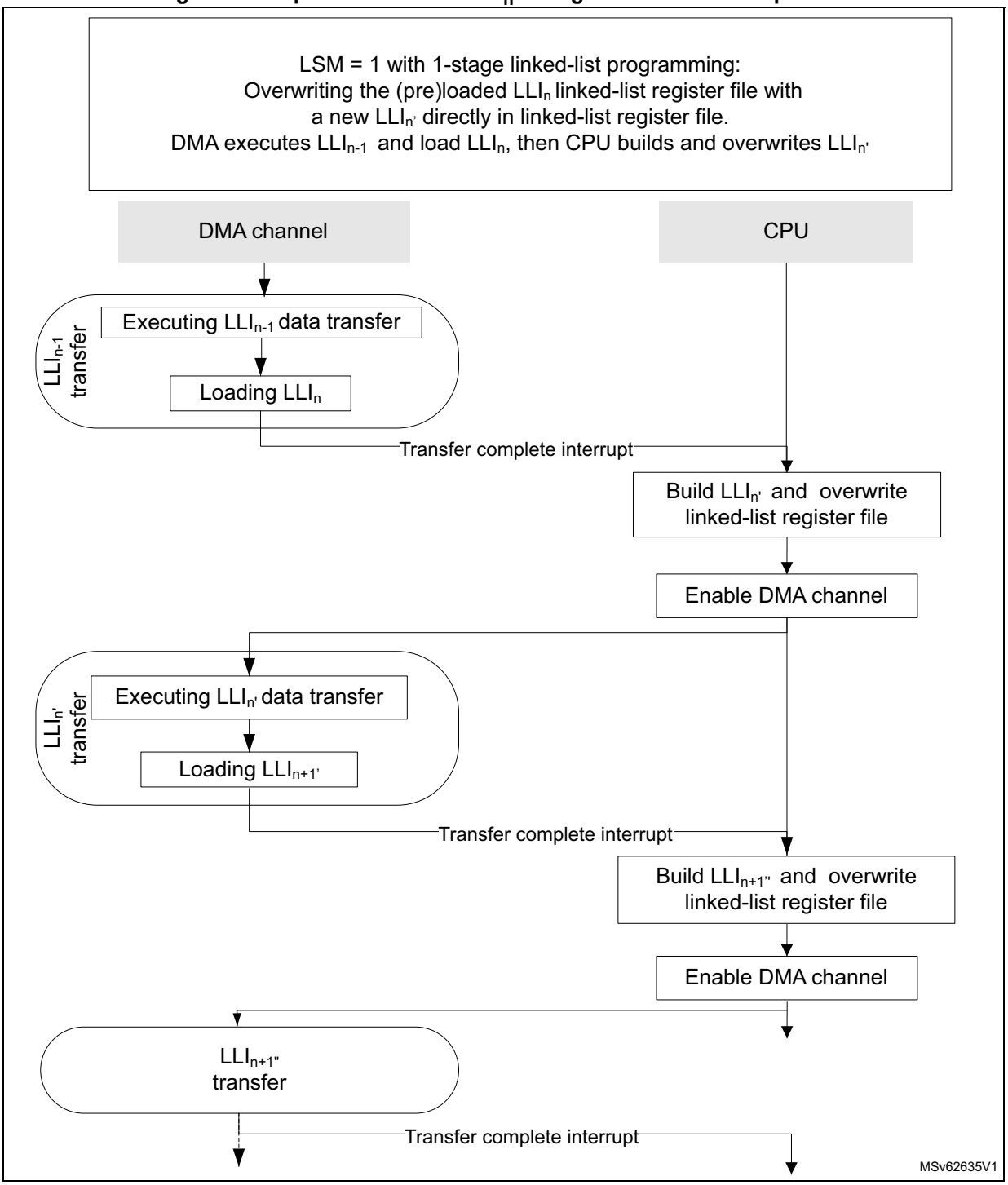
Figure 81. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 82*.

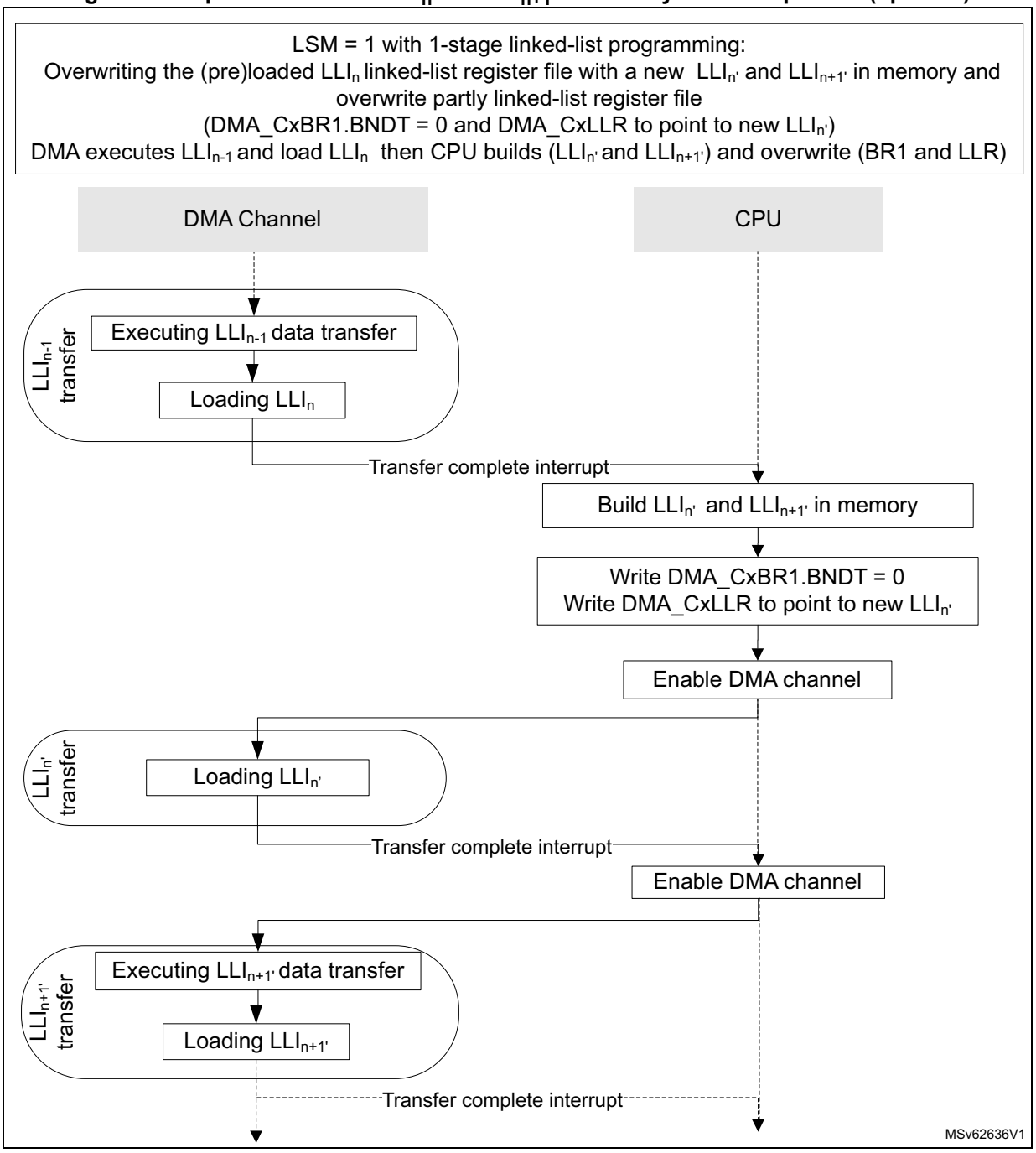
Figure 82. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in *Figure 83*.

Figure 83. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



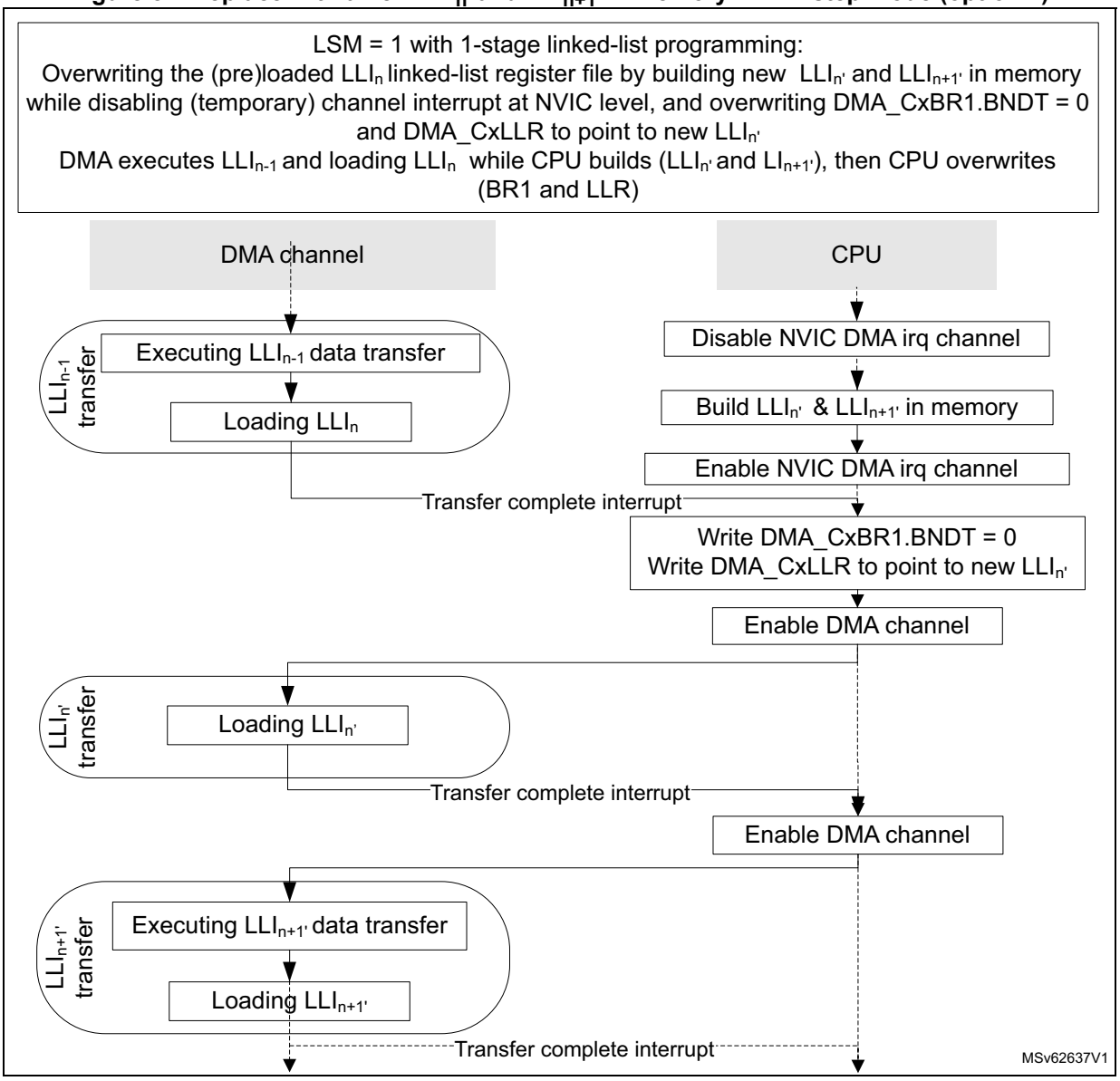
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 84*.

Figure 84. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



16.4.9 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before that this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 16.4.4*.)

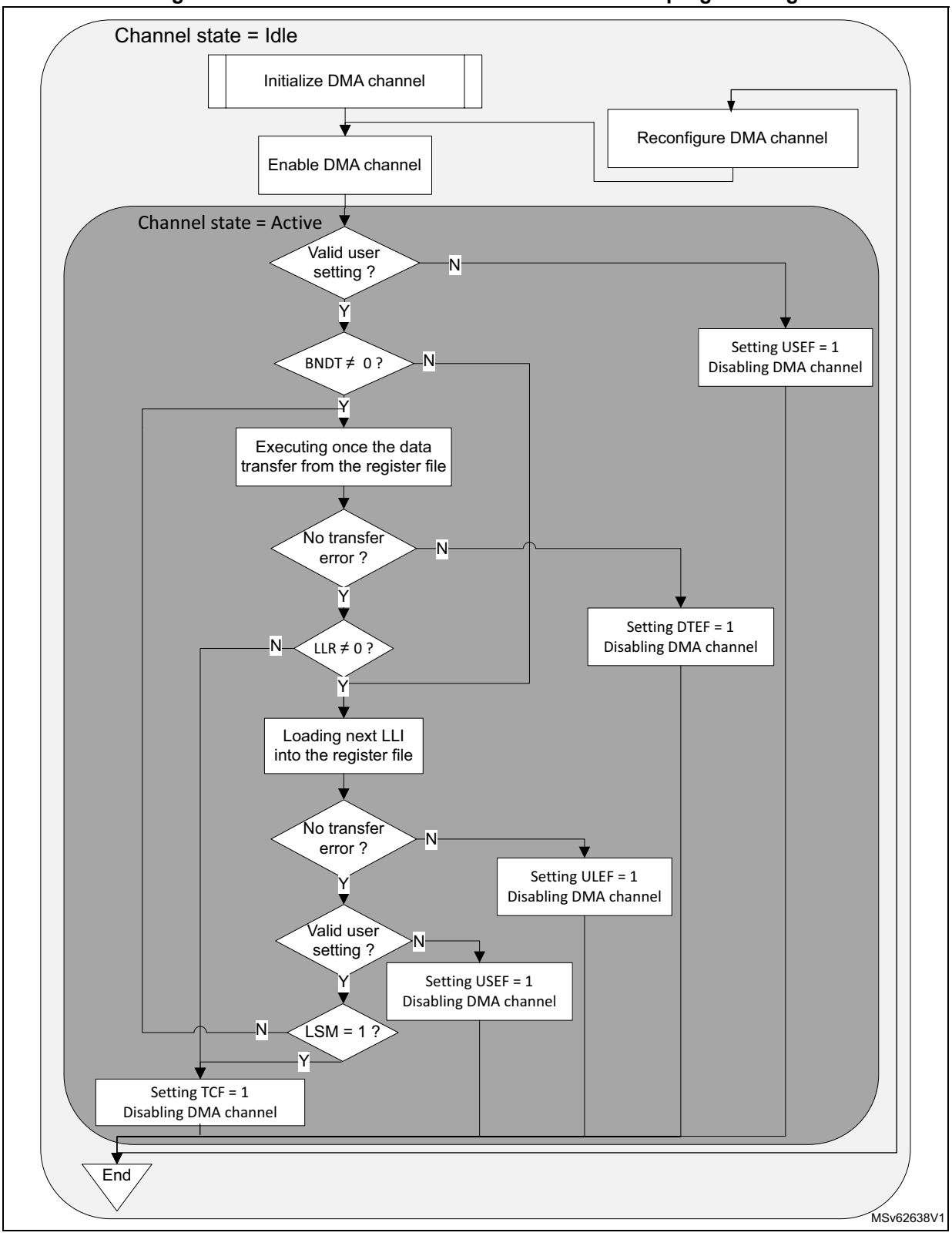
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 85 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 85 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 85, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 16.3.6).

Figure 85. GPDMA channel execution and linked-list programming



16.4.10 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 16.3.2* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 141. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

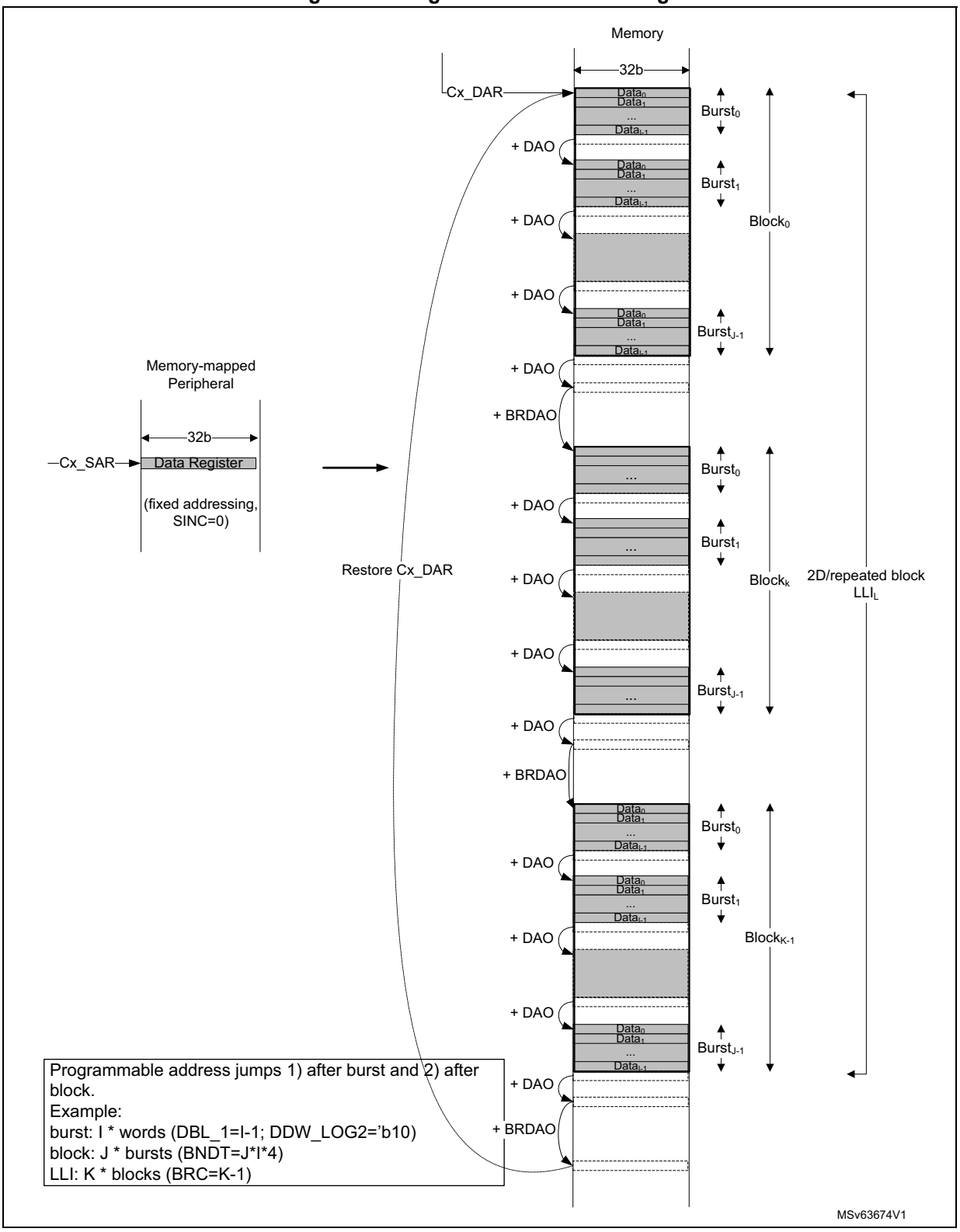
GPDMA burst with 2D addressing (channel x = 6 to 7)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 86* with a 2D destination buffer.

Figure 86. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is the half of the FIFO size of the channel x, in bytes (see *Section 16.3.2* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see *Table 142*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 142. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 142. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 142. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1		$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7,B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

16.4.11 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[7:0] field (see *Section 16.3.4*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 87*, is defined by:

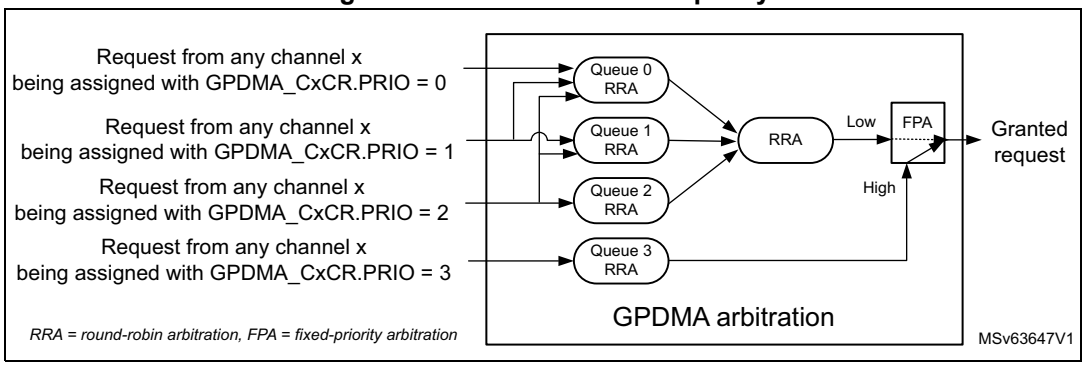
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1, or 2) for non time-sensitive channels with priority 0, 1, or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.

– Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
– Any queue 0, 1, or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
– Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1, and queue 2.

Figure 87. GPDMA arbitration policy



GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  – If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).
  $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).
  $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$
  $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).
  $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$
  $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$
  The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is
    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is
    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is
    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is
    $B_{n, P0} = 1/N *1/9 * (1 - B_{Q3})$
    $B_{n, P1} = 1/N *5/18 * (1 - B_{Q3})$
    $B_{n, P2} = 1/N *11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ ($K>0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 ($K > 0$)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 ($K = 0$)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

16.4.12 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see *Section 16.3.7* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 6 to 7), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

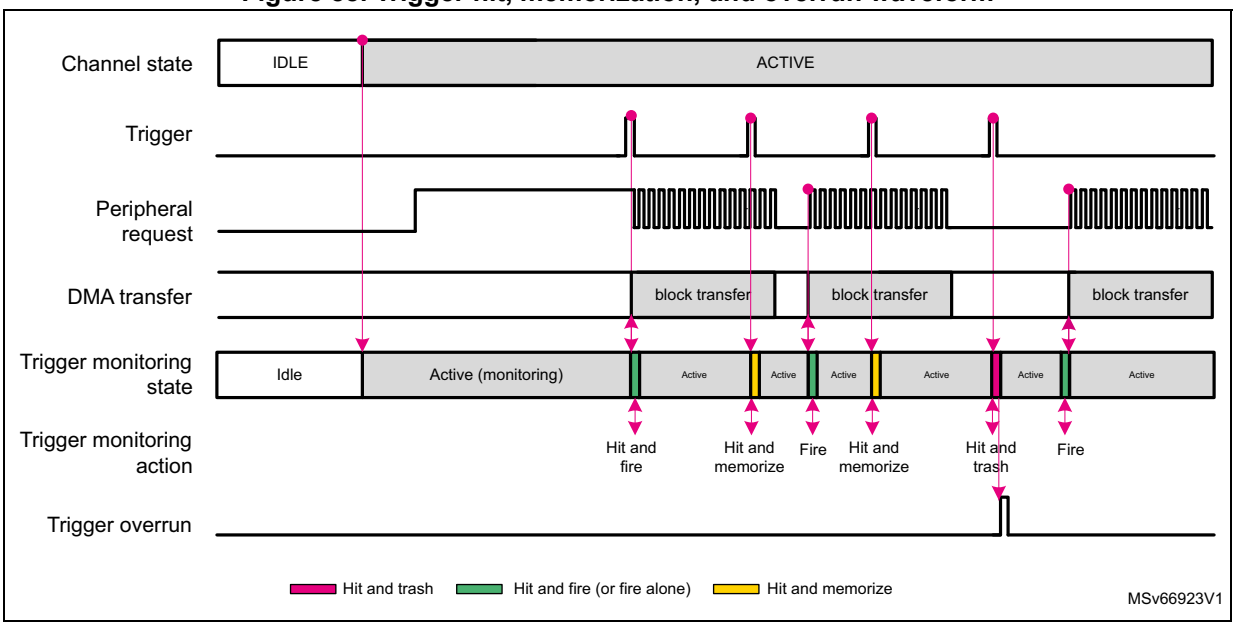
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Figure 88 illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 88. Trigger hit, memorization, and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

16.4.13 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

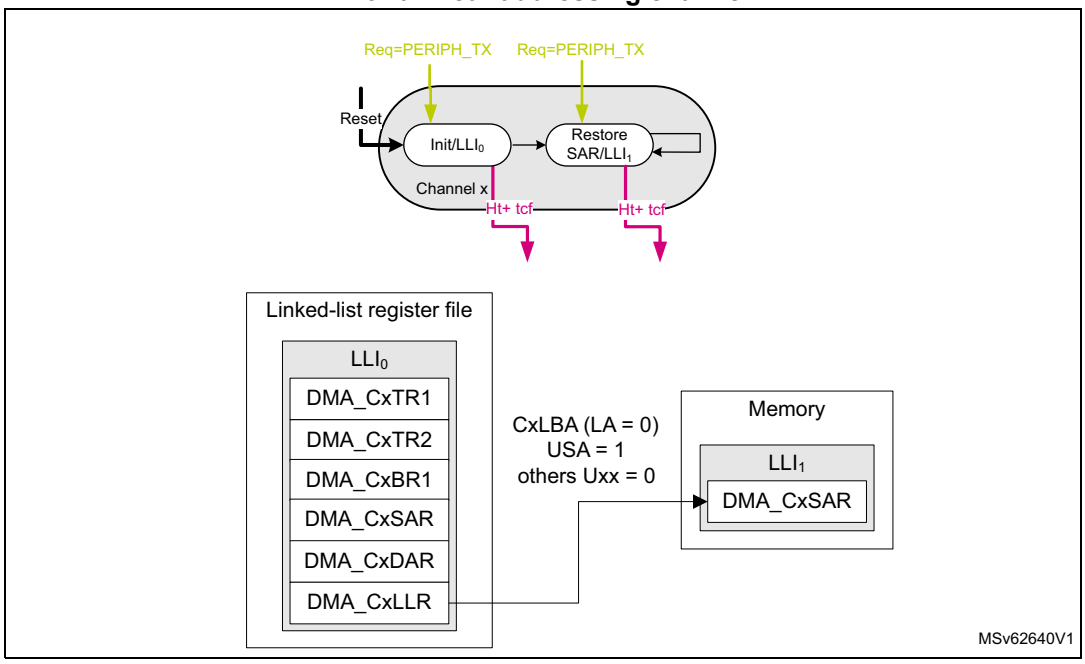
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

Figure 89 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 89. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

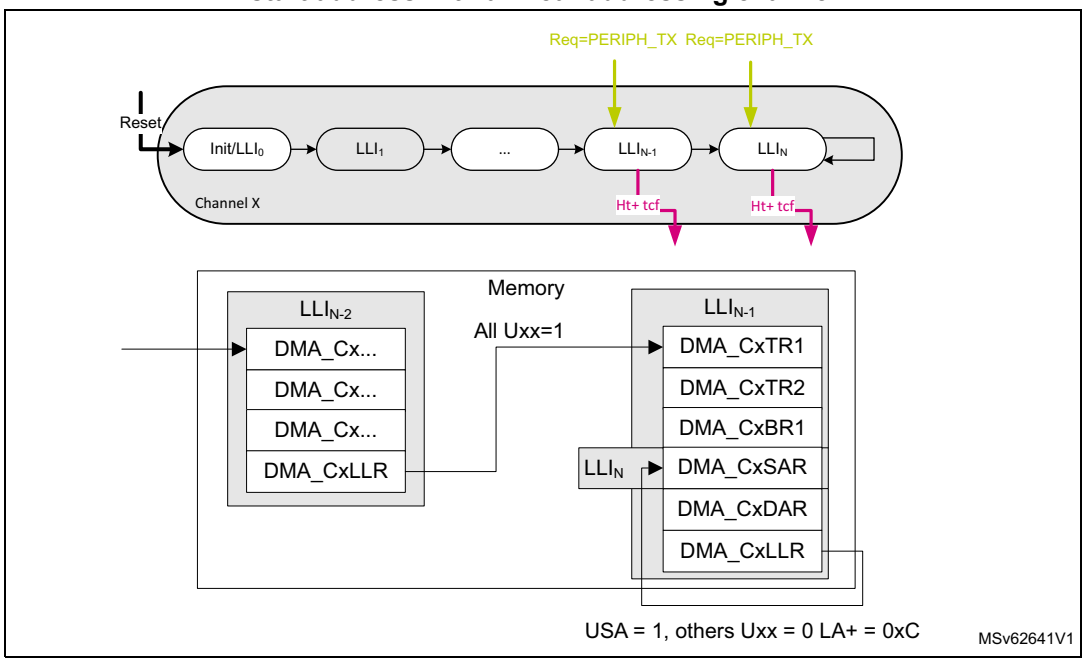


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

Figure 90 illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 90. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



16.4.14 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 16.3.6*).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with a hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

16.4.15 GPDMA secure/nonsecure channel

The GPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or a nonsecure hardware resource, as configured by GPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a register field of this channel is forced to return 0, except for GPDMA_SECCFGR, GPDMA_PRIVCFGR and GPDMA_RCFGLOCKR that are readable by a nonsecure agent.
- A nonsecure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the GPDMA data transfer from the source (GPDMA_CxTR1.SSEC) and the GPDMA data transfer to the destination (GPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with secure transfers via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a secure bus that reflects GPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each GPDMA channel x.

The GPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure GPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the nonsecure software can reprogram the GPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

16.4.16 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGR, GPDMA_SECCFGR, and GPDMA_RCFGLOCKR that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

16.4.17 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single or a burst) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a GPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 6 to 7): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])

- for channel x (x = 6 to 7): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- a programmed double-word source data width (GPDMA_CxTR1.SDW_LOG2[1:0] = 11)
- a programmed double-word destination data width (GPDMA_CxTR1.DDW_LOG2[1:0] = 11)
- a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

16.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

16.6 GPDMA in low-power modes

Table 143. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode.
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 16.3.3* to know if any Stop mode is supported.

16.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 144. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

- A half transfer event is an half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 16.4.17*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

16.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

16.8.1 GPDMA secure configuration register (GPDMA_SECCFGR)

Address offset: 0x00

Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **SECx**: secure state of channel x (x = 7 to 0)
0: Nonsecure
1: Secure

16.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure.

This register can mix secure and nonsecure information. If a channel x is configured as secure (GPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:0 **PRIVx**: privileged state of channel x (x = 7 to 0)
0: unprivileged
1: privileged

16.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attributes in order to individually lock, for example at boot time, the secure privileged attributes of any GPDMA

channel/resource (to lock the setting of GPDMA_CxSECCFGR and GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

Note: If TZEN = 0, this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
								rs	rs	rs	rs	rs	rs	rs	rs

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

Bits 7:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx and GPDMA_PRIVCFGR.PRIVx, until a global GPDMA reset (x = 7 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

16.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with GPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of GPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of GPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in GPDMA_CxFCR.

If a channel x is in secure state (GPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

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

Bits 7:0 **MISx**: masked interrupt status of channel x (x = 7 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

16.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR)

Address offset: 0x10

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (GPDMA_SECCFGR.SECx = 1). It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
								r	r	r	r	r	r	r	r

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

Bits 7:0 **MISx**: masked interrupt status of the secure channel x (x = 7 to 0)
0: no interrupt occurred on the secure channel x
1: an interrupt occurred on the secure channel x

16.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (GPDMA_SECCFGR.SECx).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

16.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

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

16.8.8 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level
Number of available write beats in the FIFO, in units of the programmed destination data width (see GPDMA_CxTR1.DDW_LOG2[1:0], in units of bytes, half-words, or words).

Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half transfer flag
0: no half transfer event
1: a half transfer event occurred
A half transfer event is either a half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).
A half block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

16.8.9 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x GPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 72*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset
This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).
The reset is effective when the channel is in steady state, meaning one of the following:
- active channel in suspended state (GPDMA_CxSR.SUSPF = 1 and GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).
After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 73*).
0: no channel reset
1: channel reset

Bit 0 **EN:** enable
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:
this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).
Else, this bit can be asserted by software.
Writing 0 into this EN bit is ignored.
0: write: ignored, read: channel disabled
1: write: enable channel, read: channel enabled

16.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or non-privileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: security attribute of the GPDMA transfer to the destination

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer to the destination is nonsecure.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

When configured in packing mode (PAM[1] = 1 and destination data width different from source data width), a source block size must be a multiple of the destination data width (see GPDMA_CxBR1.BNDT[2:0] versus DDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

Bit 15 **SSEC**: security attribute of the GPDMA transfer from the source

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer from the source is nonsecure.

Bit 14 **SAP**: source allocated port

This bit is used to allocate the master port for the source transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word
If the source data width is shorter than a word, this bit is ignored.
If the source data width is a word:
0:no byte-based exchange within the unaligned half-word of each source word
1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode
If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.
Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.
Condition: PAM_1
00: source data is transferred as right aligned, padded with 0s up to the destination data width
01: source data is transferred as right aligned, sign extended up to the destination data width
10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer
Condition: PAM_2
00: source data is transferred as right aligned, left-truncated down to the destination data width
01: source data is transferred as left-aligned, right-truncated down to the destination data width
10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63
The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst
0: fixed burst
1: contiguously incremented burst
The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: A source block size must be a multiple of the source data width (GPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

16.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]					
rw	rw					rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: channel x (x = 0 to 5), same as 00, channel x (x = 6 to 7), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x = 6 to 7), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TRIGSEL[5:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 16.3.7*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 6 to 7), for each block if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 5), same as 00; channel x (x = 6 to 7), at 2D/repeated block level. The first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 16.3.2 for the list of the implemented channels with this feature).

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see *Section 16.3.6* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).

- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 16.3.4*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[7:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1(software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 16.3.4*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

16.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

16.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is

automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI

- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

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

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.

– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.

- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

16.8.14 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 6 to 7). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

16.8.15 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 6 to 7), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

16.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

16.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

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

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

16.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 5)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw

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

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0: no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

16.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 6 to 7)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).
Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

16.8.20 GPDMA register map

Table 145. GPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPDMA_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																									0	0	0	0	0	0	0	0
0x04	GPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																									0	0	0	0	0	0	0	0
0x08	GPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																									0	0	0	0	0	0	0	0
0x0C	GPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0

Table 145. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	GPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																									0	0	0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x=0 to 7)	GPDMA_ CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x54 to 0x58+0x80*x (x=0 to 7)	Reserved	Reserved																															
0x5C+0x80 * x (x=0 to 7)	GPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x60+ 0x80 * x (x=0 to 7)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0								1
0x64+ 0x80 * x (x=0 to 7)	GPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0						0	0	0
0x68 to 0x8C+0x80 * x (x=0 to 7)	Reserved	Reserved																															
0x90+0x80 * x (x=0 to 7)	GPDMA_CxTR1	DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2 [1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2 [1:0]	
	Reset value	0	0			0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
0x94+0x80 * x (x=0 to 7)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	Res.	TRIGSEL[5:0]						TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
	Reset value	0	0					0	0			0	0	0	0	0	0	0	0		0	0	0	0		0	0		0	0	0	0	0
0x98+0x80 * x (x=0 to 5)	GPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98+0x80 * x (x=6 to 7)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C+0x80 * x (x=0 to 7)	GPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0+0x80 * x (x=0 to 7)	GPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4+0x80 * x (x=6 to 7)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8+0x80 * x (x=6 to 7)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC+0x80 * x (x=0 to 5)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xCC+0x80 * x (x=6 to 7)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

17 Nested vectored interrupt controller (NVIC)

17.1 NVIC main features

- Up to 133 maskable interrupt channels (not including the 16 Cortex-M33 with FPU interrupt lines)
- 16 programmable priority levels (four bits of interrupt priority used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

The NVIC and the processor core interface are closely coupled, enabling low-latency interrupt processing and efficient processing of late arriving interrupts.

The NVIC registers are banked across secure and non-secure states.

All interrupts. including the core exceptions, are managed by the NVIC.

17.2 SysTick calibration value register

The Cortex-M33 with TrustZone mainline security extension embeds two SysTick timers.

When TrustZone is activated, the following SysTick timers are available:

- SysTick, secure instance
- SysTick, non-secure instance

When TrustZone is disabled, only one SysTick timer is available.

The SysTick timer calibration value (STCALIB) is 0x3E8. It gives a reference time base of 1 ms based on a SysTick clock frequency of 1 MHz. To match the 1 ms time base for an application running at a given frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- When SysTick clock source is CPU clock HCLK reload value = (HCLK x STCALIB) - 1
- When SysTick clock source is external clock (HCLK/8) reload value = ((HCLK / 8) x STCALIB) - 1

The HCLK refers to the AHB frequency value in MHz.

Example: SysTick clock source is CPU clock HCLK of 100 MHz, to match a time base of 1 ms: SysTick reload value = (100 x STCALIB) - 1 = 0x1869F.

17.3 Interrupt and exception vectors

The gray rows in *Table 146* and *Table 147* describe the vectors without specific position.

Table 146. STM32H562/563/573xx vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-4	Fixed	SettableReset	Reset	0x0000 0004
-	-2	Fixed	NMI	Non maskable interrupt. The RCC clock security system (CSS) is linked to the NMI vector.	0x0000 0008
-	-3 or -1	Fixed	Secure HardFault	Secure Hard fault	0x0000 000C
-	-1	Fixed	Non-secure HardFault	Non-secure Hard fault. All classes of fault.	0x0000 000C
-	0	Settable	MemManage	Memory management	0x0000 0010
-	1	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	3	Settable	SecureFault	Secure fault	0x0000 001C
-	-	-	-	Reserved	0x0000 0020 - 0x0000 0028
-	4	-	SVC	System service call via SWI instruction	0x0000 002C
-	5	-	Debug Monitor	Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	6	Settable	PendSV	Pendable request for system service	0x0000 0038
-	7	Settable	SysTick	System tick timer	0x0000 003C
0	8	Settable	WWDG	Window watchdog interrupt	0x0000 0040
1	9	Settable	PVD_AVD	Power voltage monitor/ Analog voltage monitor	0x0000 0044
2	10	Settable	RTC	RTC global non-secure interrupts	0x0000 0048
3	11	Settable	RTC_S	RTC global secure interrupts	0x0000 004C
4	12	Settable	TAMP	Tamper global interrupts	0x0000 0050
5	13	Settable	RAMCFG	RAM configuration global interrupt	0x0000 0054
6	14	Settable	FLASH	Flash non-secure global interrupt	0x0000 0058
7	15	Settable	FLASH_S	Flash secure global interrupt	0x0000 005C
8	16	Settable	GTZC	GTZC global interrupt	0x0000 0060
9	17	Settable	RCC	RCC non-secure global interrupt	0x0000 0064
10	18	Settable	RCC_S	RCC secure global interrupt	0x0000 0068
11	19	Settable	EXTI0	EXTI Line0 interrupt	0x0000 006C
12	20	Settable	EXTI1	EXTI Line1 interrupt	0x0000 0070

Table 146. STM32H562/563/573xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
13	21	Settable	EXTI2	EXTI Line2 interrupt	0x0000 0074
14	22	Settable	EXTI3	EXTI Line3 interrupt	0x0000 0078
15	23	Settable	EXTI4	EXTI Line4 interrupt	0x0000 007C
16	24	Settable	EXTI5	EXTI Line5 interrupt	0x0000 0080
17	25	Settable	EXTI6	EXTI Line6 interrupt	0x0000 0084
18	26	Settable	EXTI7	EXTI Line7 interrupt	0x0000 0088
19	27	Settable	EXTI8	EXTI Line8 interrupt	0x0000 008C
20	28	Settable	EXTI9	EXTI Line9 interrupt	0x0000 0090
21	29	Settable	EXTI10	EXTI Line10 interrupt	0x0000 0094
22	30	Settable	EXTI11	EXTI Line11 interrupt	0x0000 0098
23	31	Settable	EXTI12	EXTI Line12 interrupt	0x0000 009C
24	32	Settable	EXTI13	EXTI Line13 interrupt	0x0000 00A0
25	33	Settable	EXTI14	EXTI Line14 interrupt	0x0000 00A4
26	34	Settable	EXTI15	EXTI Line15 interrupt	0x0000 00A8
27	35	Settable	GPDMA1_CH0	GPDMA1 channel 0 global interrupt	0x0000 00AC
28	36	Settable	GPDMA1_CH1	GPDMA1 channel 1 global interrupt	0x0000 00B0
29	37	Settable	GPDMA1_CH2	GPDMA1 channel 2 global interrupt	0x0000 00B4
30	38	Settable	GPDMA1_CH3	GPDMA1 channel 3 global interrupt	0x0000 00B8
31	39	Settable	GPDMA1_CH4	GPDMA1 channel 4 global interrupt	0x0000 00BC
32	40	Settable	GPDMA1_CH5	GPDMA1 channel 5 global interrupt	0x0000 00C0
33	41	Settable	GPDMA1_CH6	GPDMA1 channel 6 global interrupt	0x0000 00C4
34	42	Settable	GPDMA1_CH7	GPDMA1 channel 7 global interrupt	0x0000 00C8
35	43	Settable	IWDG	Independent watchdog interrupt	0x0000 00CC
36	44	Settable	SAES(1)	Secure AES	0x0000 00D0
37	45	Settable	ADC1	ADC1 global interrupt	0x0000 00D4
38	46	Settable	DAC1	DAC1 global interrupt	0x0000 00D8
39	47	Settable	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 00DC
40	48	Settable	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 00E0
41	49	Settable	TIM1_BRK/TIM1_TERR/ TIM1_IERR	TIM1 break/transition error/ index error	0x0000 00E4
42	50	Settable	TIM1_UP	TIM1 update	0x0000 00E8
43	51	Settable	TIM1_TRG_COM/ TIM1_DIR/TIM1_IDX	TIM1 trigger and commutation/direction change interrupt/index	0x0000 00EC
44	52	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00F0

Table 146. STM32H562/563/573xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
45	53	Settable	TIM2	TIM2 global interrupt	0x0000 00F4
46	54	Settable	TIM3	TIM3 global interrupt	0x0000 00F8
47	55	Settable	TIM4	TIM4 global interrupt	0x0000 00FC
48	56	Settable	TIM5	TIM5 global interrupt	0x0000 0100
49	57	Settable	TIM6	TIM6 global interrupt	0x0000 0104
50	58	Settable	TIM7	TIM7 global interrupt	0x0000 0108
51	59	Settable	I2C1_EV	I2C1 event interrupt	0x0000 010C
52	60	Settable	I2C1_ER	I2C1 error interrupt	0x0000 0110
53	61	Settable	I2C2_EV	I2C2 event interrupt	0x0000 0114
54	62	Settable	I2C2_ER	I2C2 error interrupt	0x0000 0118
55	63	Settable	SPI1	SPI1 global interrupt	0x0000 011C
56	64	Settable	SPI2	SPI2 global interrupt	0x0000 0120
57	65	Settable	SPI3	SPI3 global interrupt	0x0000 0124
58	66	Settable	USART1	USART1 global interrupt	0x0000 0128
59	67	Settable	USART2	USART2 global interrupt	0x0000 012C
60	68	Settable	USART3	USART3 global interrupt	0x0000 0130
61	69	Settable	UART4	UART4 global interrupt	0x0000 0134
62	70	Settable	UART5	UART5 global interrupt	0x0000 0138
63	71	Settable	LPUART1	LPUART1 global interrupt or R wake-up or T wake-up through EXTI line	0x0000 013C
64	72	Settable	LPTIM1 or LPTIM1_AIT	LPTIM1 global interrupt or AIT through EXTI line	0x0000 0140
65	73	Settable	TIM8_BRK/TIM8_TERR/ TIM8_IERR	TIM8 break interrupt/transition error/index error	0x0000 0144
66	74	Settable	TIM8_UP	TIM8 update interrupt	0x0000 0148
67	75	Settable	TIM8_TRG_COM/ TIM8_DIR/TIM8_IDX	TIM8 trigger and commutation interrupt/ direction change interrupt/index	0x0000 014C
68	76	Settable	TIM8_CC	TIM8 capture compare interrupt	0x0000 0150
69	77	Settable	ADC2	ADC2 global interrupt	0x0000 0154
70	78	Settable	LPTIM2 or LPTIM2_AIT	LPTIM2 global interrupt or AIT through EXTI line	0x0000 0158
71	79	Settable	TIM15	TIM15 global interrupt	0x0000 015C
72	80	Settable	TIM16	TIM16 global interrupt	0x0000 0160
73	81	Settable	TIM17	TIM17 global interrupt	0x0000 0164
74	82	Settable	USB_FS	USB FS global interrupt	0x0000 0168

Table 146. STM32H562/563/573xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
75	83	Settable	CRS	Clock recovery system global interrupt	0x0000 016C
76	84	Settable	UCPD1	UCPD1 global interrupt	0x0000 0170
77	85	Settable	FMC	FMC global interrupt	0x0000 0174
78	86	Settable	OCTOSPI1	OCTOSPI1 global interrupt	0x0000 0178
79	87	Settable	SDMMC1	SDMMC1 global interrupt	0x0000 017C
80	88	Settable	I2C3_EV	I2C3 event interrupt	0x0000 0180
81	89	Settable	I2C3_ER	I2C3 error interrupt	0x0000 0184
82	90	Settable	SPI4	SPI4 global interrupt	0x0000 0188
83	91	Settable	SPI5	SPI5 global interrupt	0x0000 018C
84	92	Settable	SPI6	SPI6 global interrupt	0x0000 0190
85	93	Settable	USART6	USART6 global interrupt	0x0000 0194
86	94	Settable	USART10	USART10 global interrupt	0x0000 0198
87	95	Settable	USART11	USART11 global interrupt	0x0000 019C
88	96	Settable	SAI1	SAI1 global interrupt	0x0000 01A0
89	97	Settable	SAI2	SAI2 global interrupt	0x0000 01A4
90	98	Settable	GPDMA2_CH0	GPDMA2 channel0 global interrupt	0x0000 01A8
91	99	Settable	GPDMA2_CH1	GPDMA2 channel1 global interrupt	0x0000 01AC
92	100	Settable	GPDMA2_CH2	GPDMA2 channel2 global interrupt	0x0000 01B0
93	101	Settable	GPDMA2_CH3	GPDMA2 channel3 global interrupt	0x0000 01B4
94	102	Settable	GPDMA2_CH4	GPDMA2 channel4 global interrupt	0x0000 01B8
95	103	Settable	GPDMA2_CH5	GPDMA2 channel5 global interrupt	0x0000 01BC
96	104	Settable	GPDMA2_CH6	GPDMA2 channel6 global interrupt	0x0000 01C0
97	105	Settable	GPDMA2_CH7	GPDMA2 channel7 global interrupt	0x0000 01C4
98	106	Settable	UART7	UART7 global interrupt	0x0000 01C8
99	107	Settable	UART8	UART8 global interrupt	0x0000 01CC
100	108	Settable	UART9	UART9 global interrupt	0x0000 01D0
101	109	Settable	UART12	UART12 global interrupt	0x0000 01D4
102	110	Settable	SDMMC2(2)	SDMMC2 global interrupt	0x0000 01D8
103	111	Settable	FPU	Floating point interrupt	0x0000 01DC
104	112	Settable	ICACHE	Instruction cache global interrupt	0x0000 01E0
105	113	Settable	DCACHE	Data cache global interrupt	0x0000 01E4
106	114	Settable	ETH(2)	Ethernet interrupt	0x0000 01E8
107	115	Settable	ETH_WKUP	Ethernet wake-up interrupt through EXTI line	0x0000 01EC

Table 146. STM32H562/563/573xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
108	116	Settable	DCMI_PSSI	DCMI/PSSI global interrupt	0x0000 01F0
109	117	Settable	FDCAN2_IT0(2)	FDCAN2 interrupt 0	0x0000 01F4
110	118	Settable	FDCAN2_IT1(2)	FDCAN2 interrupt 1	0x0000 01F8
111	119	Settable	CORDIC	CORDIC interrupt	0x0000 01FC
112	120	Settable	FMAC	FMAC interrupt	0x0000 0200
113	121	Settable	DTS or DTS_WKUP	DTS interrupt or AIT through EXTI line	0x0000 0204
114	122	Settable	RNG	RNG global interrupt	0x0000 0208
115	123	Settable	OTFDEC1	OTFDEC1 secure global interrupt	0x0000 020C
116	124	Settable	AES(1)	AES global interrupt	0x0000 0210
117	125	Settable	HASH	HASH interrupt	0x0000 0214
118	126	Settable	PKA	PKA global interrupt	0x0000 0218
119	127	Settable	CEC	HDMI-CEC global interrupt	0x0000 021C
120	128	Settable	TIM12	TIM12 global interrupt	0x0000 0220
121	129	Settable	TIM13	TIM13 global interrupt	0x0000 0224
122	130	Settable	TIM14	TIM14 global interrupt	0x0000 0228
123	131	Settable	I3C1_EV	I3C1 event interrupt	0x0000 022C
124	132	Settable	I3C1_ER	I3C1 error interrupt	0x0000 0230
125	133	Settable	I2C4_EV	I2C4 event interrupt	0x0000 0234
126	134	Settable	I2C4_ER	I2C4 error interrupt	0x0000 0238
127	135	Settable	LPTIM3 or LPTIM3_AIT	LPTIM3 global interrupt or AIT through EXTI line	0x0000 023C
128	136	Settable	LPTIM4 or LPTIM4_AIT	LPTIM4 global interrupt or AIT through EXTI line	0x0000 0240
129	137	Settable	LPTIM5 or LPTIM5_AIT	LPTIM5 global interrupt or AIT through EXTI line	0x0000 0244
130	138	Settable	LPTIM6 or LPTIM6_AIT	LPTIM6 global interrupt or AIT through EXTI line	0x0000 0248

1. Not available on STM32H562/563 devices.
2. Not available on STM32H562 devices.

Table 147. STM32H523/533xx vector table

Position	Priority	Type of priority	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-4	Fixed	SettableReset	Reset	0x0000 0004
-	-2	Fixed	NMI	Non maskable interrupt. The RCC clock security system (CSS) is linked to the NMI vector.	0x0000 0008
-	-3 or -1	Fixed	Secure HardFault	Secure Hard fault	0x0000 000C
-	-1	Fixed	Non-secure HardFault	Non-secure Hard fault. All classes of fault.	0x0000 000C
-	0	Settable	MemManage	Memory management	0x0000 0010
-	1	Settable	BusFault	Pre-fetch fault, memory access fault	0x0000 0014
-	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	3	Settable	SecureFault	Secure fault	0x0000 001C
-	-	-	-	Reserved	0x0000 0020 - 0x0000 0028
-	4	-	SVC	System service call via SWI instruction	0x0000 002C
-	5	-	Debug Monitor	Monitor	0x0000 0030
-	-	-	-	Reserved	0x0000 0034
-	6	Settable	PendSV	Pendable request for system service	0x0000 0038
-	7	Settable	SysTick	System tick timer	0x0000 003C
0	8	Settable	WWDG	Window watchdog interrupt	0x0000 0040
1	9	Settable	PVD_AVD	Power voltage monitor/ Analog voltage monitor	0x0000 0044
2	10	Settable	RTC	RTC global non-secure interrupts	0x0000 0048
3	11	Settable	RTC_S	RTC global secure interrupts	0x0000 004C
4	12	Settable	TAMP	Tamper global interrupts	0x0000 0050
5	13	Settable	RAMCFG	RAM configuration global interrupt	0x0000 0054
6	14	Settable	FLASH	Flash non-secure global interrupt	0x0000 0058
7	15	Settable	FLASH_S	Flash secure global interrupt	0x0000 005C
8	16	Settable	GTZC	GTZC global interrupt	0x0000 0060
9	17	Settable	RCC	RCC non-secure global interrupt	0x0000 0064
10	18	Settable	RCC_S	RCC secure global interrupt	0x0000 0068
11	19	Settable	EXTI0	EXTI Line0 interrupt	0x0000 006C
12	20	Settable	EXTI1	EXTI Line1 interrupt	0x0000 0070
13	21	Settable	EXTI2	EXTI Line2 interrupt	0x0000 0074
14	22	Settable	EXTI3	EXTI Line3 interrupt	0x0000 0078

Table 147. STM32H523/533xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
15	23	Settable	EXTI4	EXTI Line4 interrupt	0x0000 007C
16	24	Settable	EXTI5	EXTI Line5 interrupt	0x0000 0080
17	25	Settable	EXTI6	EXTI Line6 interrupt	0x0000 0084
18	26	Settable	EXTI7	EXTI Line7 interrupt	0x0000 0088
19	27	Settable	EXTI8	EXTI Line8 interrupt	0x0000 008C
20	28	Settable	EXTI9	EXTI Line9 interrupt	0x0000 0090
21	29	Settable	EXTI10	EXTI Line10 interrupt	0x0000 0094
22	30	Settable	EXTI11	EXTI Line11 interrupt	0x0000 0098
23	31	Settable	EXTI12	EXTI Line12 interrupt	0x0000 009C
24	32	Settable	EXTI13	EXTI Line13 interrupt	0x0000 00A0
25	33	Settable	EXTI14	EXTI Line14 interrupt	0x0000 00A4
26	34	Settable	EXTI15	EXTI Line15 interrupt	0x0000 00A8
27	35	Settable	GPDMA1_CH0	GPDMA1 channel 0 global interrupt	0x0000 00AC
28	36	Settable	GPDMA1_CH1	GPDMA1 channel 1 global interrupt	0x0000 00B0
29	37	Settable	GPDMA1_CH2	GPDMA1 channel 2 global interrupt	0x0000 00B4
30	38	Settable	GPDMA1_CH3	GPDMA1 channel 3 global interrupt	0x0000 00B8
31	39	Settable	GPDMA1_CH4	GPDMA1 channel 4 global interrupt	0x0000 00BC
32	40	Settable	GPDMA1_CH5	GPDMA1 channel 5 global interrupt	0x0000 00C0
33	41	Settable	GPDMA1_CH6	GPDMA1 channel 6 global interrupt	0x0000 00C4
34	42	Settable	GPDMA1_CH7	GPDMA1 channel 7 global interrupt	0x0000 00C8
35	43	Settable	IWDG	Independent watchdog interrupt	0x0000 00CC
36	44	Settable	SAES(1)	Secure AES	0x0000 00D0
37	45	Settable	ADC1	ADC1 global interrupt	0x0000 00D4
38	46	Settable	DAC1	DAC1 global interrupt	0x0000 00D8
39	47	Settable	FDCAN1_IT0	FDCAN1 Interrupt 0	0x0000 00DC
40	48	Settable	FDCAN1_IT1	FDCAN1 Interrupt 1	0x0000 00E0
41	49	Settable	TIM1_BRK/TIM1_TERR/ TIM1_IERR	TIM1 break/transition error/ index error	0x0000 00E4
42	50	Settable	TIM1_UP	TIM1 update	0x0000 00E8
43	51	Settable	TIM1_TRG_COM/ TIM1_DIR/TIM1_IDX	TIM1 trigger and commutation/direction change interrupt/index	0x0000 00EC
44	52	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 00F0
45	53	Settable	TIM2	TIM2 global interrupt	0x0000 00F4
46	54	Settable	TIM3	TIM3 global interrupt	0x0000 00F8

Table 147. STM32H523/533xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
47	55	Settable	TIM4	TIM4 global interrupt	0x0000 00FC
48	56	Settable	TIM5	TIM5 global interrupt	0x0000 0100
49	57	Settable	TIM6	TIM6 global interrupt	0x0000 0104
50	58	Settable	TIM7	TIM7 global interrupt	0x0000 0108
51	59	Settable	I2C1_EV	I2C1 event interrupt	0x0000 010C
52	60	Settable	I2C1_ER	I2C1 error interrupt	0x0000 0110
53	61	Settable	I2C2_EV	I2C2 event interrupt	0x0000 0114
54	62	Settable	I2C2_ER	I2C2 error interrupt	0x0000 0118
55	63	Settable	SPI1	SPI1 global interrupt	0x0000 011C
56	64	Settable	SPI2	SPI2 global interrupt	0x0000 0120
57	65	Settable	SPI3	SPI3 global interrupt	0x0000 0124
58	66	Settable	USART1	USART1 global interrupt	0x0000 0128
59	67	Settable	USART2	USART2 global interrupt	0x0000 012C
60	68	Settable	USART3	USART3 global interrupt	0x0000 0130
61	69	Settable	UART4	UART4 global interrupt	0x0000 0134
62	70	Settable	UART5	UART5 global interrupt	0x0000 0138
63	71	Settable	LPUART1	LPUART1 global interrupt or R wake-up or T wake-up through EXTI line	0x0000 013C
64	72	Settable	LPTIM1 or LPTIM1_AIT	LPTIM1 global interrupt or AIT through EXTI line	0x0000 0140
65	73	Settable	TIM8_BRK/TIM8_TERR/ TIM8_IERR	TIM8 break interrupt/transition error/index error	0x0000 0144
66	74	Settable	TIM8_UP	TIM8 update interrupt	0x0000 0148
67	75	Settable	TIM8_TRG_COM/ TIM8_DIR/TIM8_IDX	TIM8 trigger and commutation interrupt/ direction change interrupt/index	0x0000 014C
68	76	Settable	TIM8_CC	TIM8 capture compare interrupt	0x0000 0150
69	77	Settable	ADC2	ADC2 global interrupt	0x0000 0154
70	78	Settable	LPTIM2 or LPTIM2_AIT	LPTIM2 global interrupt or AIT through EXTI line	0x0000 0158
71	79	Settable	TIM15	TIM15 global interrupt	0x0000 015C
72	80	-	-	Reserved	0x0000 0164
73	81	-	-	Reserved	0x0000 0168
74	82	Settable	USB_FS	USB FS global interrupt	0x0000 0168
75	83	Settable	CRS	Clock recovery system global interrupt	0x0000 016C
76	84	Settable	UCPD1	UCPD1 global interrupt	0x0000 0170

Table 147. STM32H523/533xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
77	85	Settable	FMC	FMC global interrupt	0x0000 0174
78	86	Settable	OCTOSPI1	OCTOSPI1 global interrupt	0x0000 0178
79	87	Settable	SDMMC1	SDMMC1 global interrupt	0x0000 017C
80	88	Settable	I2C3_EV	I2C3 event interrupt	0x0000 0180
81	89	Settable	I2C3_ER	I2C3 error interrupt	0x0000 0184
82	90	Settable	SPI4	SPI4 global interrupt	0x0000 0188
83	91	-	-	Reserved	0x0000 018C
84	92	-	-	Reserved	0x0000 0190
85	93	Settable	USART6	USART6 global interrupt	0x0000 0194
86	94	-	-	Reserved	0x0000 0198
87	95	-	-	Reserved	0x0000 019C
88	96	-	-	Reserved	0x0000 01A0
89	97	-	-	Reserved	0x0000 01A4
90	98	Settable	GPDMA2_CH0	GPDMA2 channel0 global interrupt	0x0000 01A8
91	99	Settable	GPDMA2_CH1	GPDMA2 channel1 global interrupt	0x0000 01AC
92	100	Settable	GPDMA2_CH2	GPDMA2 channel2 global interrupt	0x0000 01B0
93	101	Settable	GPDMA2_CH3	GPDMA2 channel3 global interrupt	0x0000 01B4
94	102	Settable	GPDMA2_CH4	GPDMA2 channel4 global interrupt	0x0000 01B8
95	103	Settable	GPDMA2_CH5	GPDMA2 channel5 global interrupt	0x0000 01BC
96	104	Settable	GPDMA2_CH6	GPDMA2 channel6 global interrupt	0x0000 01C0
97	105	Settable	GPDMA2_CH7	GPDMA2 channel7 global interrupt	0x0000 01C4
98	106	-	-	Reserved	0x0000 01C8
99	107	-	-	Reserved	0x0000 01CC
100	108	-	-	Reserved	0x0000 01D0
101	109	-	-	Reserved	0x0000 01D4
102	110	-	-	Reserved	0x0000 01D8
103	111	Settable	FPU	Floating point interrupt	0x0000 01DC
104	112	Settable	ICACHE	Instruction cache global interrupt	0x0000 01E0
105	113	Settable	DCACHE	Data cache global interrupt	0x0000 01E4
106	114	-	-	Reserved	0x0000 01E8
107	115	-	-	Reserved	0x0000 01EC
108	116	Settable	DCMI_PSSI	DCMI/PSSI global interrupt	0x0000 01F0
109	117	Settable	FDCAN2_IT0	FDCAN2 interrupt 0	0x0000 01F4

Table 147. STM32H523/533xx vector table (continued)

Position	Priority	Type of priority	Acronym	Description	Address
110	118	Settable	FDCAN2_IT1	FDCAN2 interrupt 1	0x0000 01F8
111	119	-	-	Reserved	0x0000 01FC
112	120	-	-	Reserved	0x0000 0200
113	121	Settable	DTS or DTS_WKUP	DTS interrupt or AIT through EXTI line	0x0000 0204
114	122	Settable	RNG	RNG global interrupt	0x0000 0208
115	123	Settable	OTFDEC1	OTFDEC1 secure global interrupt	0x0000 020C
116	124	Settable	AES(1)	AES global interrupt	0x0000 0210
117	125	Settable	HASH	HASH interrupt	0x0000 0214
118	126	Settable	PKA(1)	PKA global interrupt	0x0000 0218
119	127	Settable	CEC	HDMI-CEC global interrupt	0x0000 021C
120	128	Settable	TIM12	TIM12 global interrupt	0x0000 0220
121	129	-	-	Reserved	0x0000 0224
122	130	-	-	Reserved	0x0000 0228
123	131	Settable	I3C1_EV	I3C1 event interrupt	0x0000 022C
124	132	Settable	I3C1_ER	I3C1 error interrupt	0x0000 0230
125	133	-	-	Reserved	0x0000 0234
126	134	-	-	Reserved	0x0000 0238
127	135	-	-	Reserved	0x0000 023C
128	136	-	-	Reserved	0x 0000 0240
129	137	-	-	Reserved	0x0000 0244
130	138	-	-	Reserved	0x0000 0248
131	139	Settable	I3C2_EV	I3C2 event interrupt	0x0000 024C
132	140	Settable	I3C2_ER	I3C2 error interrupt	0x0000 0250
133	141	-	-	Reserved	0x0000 0254

1. Not available on STM32H523 devices.

18 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the individual CPU and system wake-up through configurable and direct event inputs. It provides wake-up requests to the power control and generates an interrupt request to the CPU NVIC and events to the CPU event input. For the CPU, an additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wake-up requests allow the system to be woken up from Stop modes.

The interrupt request and event request generation can be used also in Run modes.

The EXTI also includes the EXTI mux IO port selection.

18.1 EXTI main features

The EXTI main features are the following:

- Up to 59 input events supported
- All event inputs allow the possibility to wake up the system.
- Events that do not have an associated wake-up flag in the peripheral, have a flag in the EXTI and generate an interrupt to the CPU from the EXTI.
- Events can be used to generate a CPU wake-up event.

The asynchronous event inputs are classified in two groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse), with the following features
  - Selectable active trigger edge
  - Interrupt pending status register bits independent for the rising and falling edge
  - Individual interrupt and event generation mask, used for conditioning the CPU wake-up, interrupt and event generation
  - Software trigger possibility
  - Secure events: The access to control and configuration bits of secure input events can be made secure and or privilege.
  - EXTI IO port selection
- Direct events (interrupt and wake-up sources from peripherals having an associated flag which requires to be cleared in the peripheral), with the following features:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask, used to condition the CPU wake-up and event generation
  - No software trigger possibility

18.2 EXTI block diagram

The EXTI consists of a register block accessed via an AHB interface, the event input trigger block, the masking block and EXTI mux as shown in *Figure 91*.

The register block contains all the EXTI registers.

The event input trigger block provides event input edge trigger logic.

The masking block provides the event input distribution to the different wake-up, interrupt and event outputs, and their masking.

The EXTI mux provides the IO port selection on to the EXTI event signal.

Figure 91. EXTI block diagram

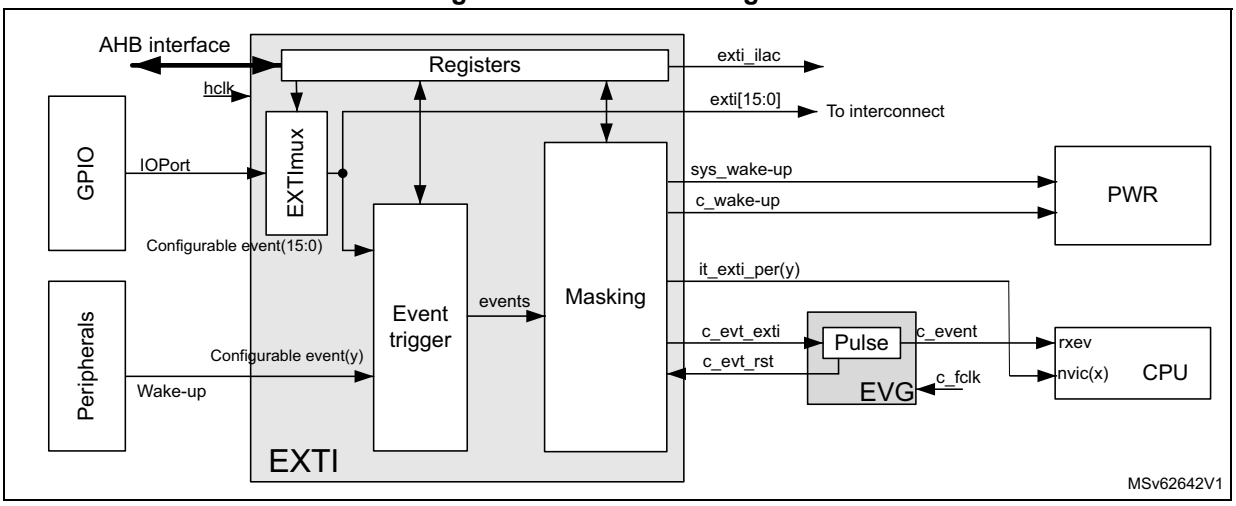


Table 148. EXTI signals

Pin name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to enable security, the AHB interface supports secure accesses.
hclk	I	AHB bus clock and EXTI system clock
Configurable event(y)	I	Asynchronous wake-up events from peripherals without an associated interrupt and flag
Direct event(x)	I	Synchronous and asynchronous wake-up events from peripherals with an associated interrupt and flag
exti_ilac	O	Illegal access event
IOPort(n)	I	GPIOs block IO ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals
it_exti_per (y)	O	Interrupts to the CPU associated with configurable event (y)
c_evt_exti	O	High-level sensitive event output for CPU, synchronous to hclk
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wake-up request to PWR for ck_sys and hclk
c_wakeup	O	Wake-up request to PWR for CPU, synchronous to hclk

Table 149. EVG signals

Pin name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

18.2.1 EXTI connections between peripherals and CPU

Some peripherals able to generate wake-up or interrupt events when the system is in Stop mode, are connected to the EXTI.

- Peripheral wake-up signals that generate a pulse or do not have an interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit to be cleared. It is the EXTI interrupt, associated with the status bit, that interrupts the CPU.
- Peripheral interrupt and wake-up signals with a status bit in the peripheral to be cleared are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.

All GPIO ports input to the EXTI multiplexer allow the selection of a port pin to wake up the system via a configurable event.

The EXTI configurable event interrupts are connected to the NVIC.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

The EXTI CPU wake-up signals are connected to the PWR and are used to wake up the system and the CPU sub-system bus clocks.

18.2.2 EXTI interrupt/event mapping

The EXTI lines are connected as shown in the following table.

Table 150. EXTI line connections

EXTI Line	Line source	Line type
0-15	GPIO	Configurable
16	PVD/AVD output	Configurable
17	RTC non-secure	-
18	RTC secure	-
19	TAMP non-secure	-
20	TAMP secure	-
21	I2C1 wake-up	-
22	I2C2 wake-up	-
23	I2C3 wake-up	-
24	I3C1 wake-up	-

Table 150. EXTI line connections (continued)

EXTI Line	Line source	Line type
25	USART1 wake-up	-
26	USART2 wake-up	-
27	USART3 wake-up	-
28	UART4 wake-up	-
29	UART5 wake-up	-
30	USART6 wake-up	-
31	UART7 wake-up(1)	-
32	UART8 wake-up(1)	-
33	UART9 wake-up(1)	-
34	USART10 wake-up(1)	-
35	USART11 wake-up(1)	-
36	UART12 wake-up(1)	-
37	LPUART1 wake-up	-
38	LPTIM1	-
39	LPTIM2	-
40	SPI1 wake-up	-
41	SPI2 wake-up	-
42	SPI3 wake-up	-
43	SPI4 wake-up	-
44	SPI5 wake-up(1)	-
45	SPI6 wake-up(1)	-
46	ETH wake-up(1)	Configurable
47	USB FS wake-up	-
48	USBPD1 wake-up	-
49	LPTIM2 CH1	-
50	DTS wake-up	Configurable
51	HDMI-CEC wake-up	-
52	I2C4 wake-up(1)	-
53	V_{DDIO2} monitoring	Configurable
54	LPTIM3(1)	-
55	LPTIM4(1)	-
56	LPTIM5(1)	-
57	LPTIM6(1)	-
58	I3C2(2)	-

1. Not available on STM32H523/33xx devices.

2. Available only on STM32H523/33xx devices.

18.3 EXTI functional description

The events features are controlled from register bits as follows:

- Active trigger edge enable
  - by rising edge selection in the *EXTI rising trigger selection register (EXTI_RTSR1)* and *EXTI rising trigger selection register 2 (EXTI_RTSR2)*
  - by falling edge selection in the *EXTI falling trigger selection register (EXTI_FTSR1)* and *EXTI falling trigger selection register 2 (EXTI_FTSR2)*
- Software trigger in the *EXTI software interrupt event register (EXTI_SWIER1)* and *EXTI software interrupt event register 2 (EXTI_SWIER2)*
- Interrupt pending flag in the

  EXTI rising edge pending register (EXTI_RPR1) and *EXTI rising edge pending register 2 (EXTI_RPR2)*

  EXTI falling edge pending register (EXTI_FPR1) and *EXTI falling edge pending register 2 (EXTI_FPR2)*
- CPU wake-up and interrupt enable in the

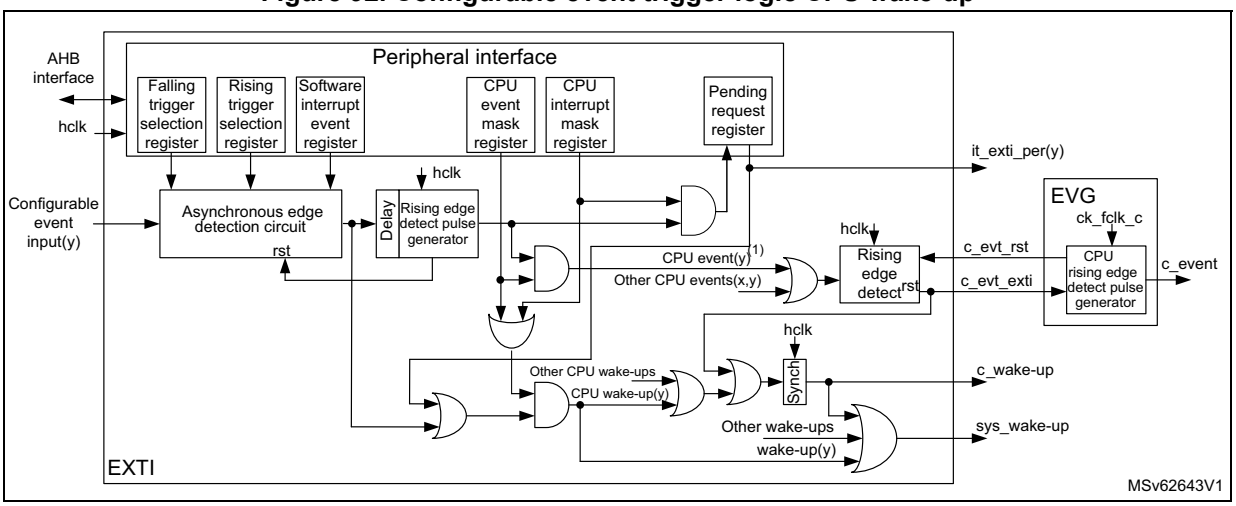
  EXTI CPU wake-up with interrupt mask register (EXTI_IMR1) and *EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2)*
- CPU wake-up and event enable

  EXTI CPU wake-up with event mask register (EXTI_EMR1) and *EXTI CPU wake-up with event mask register 2 (EXTI_EMR2)*

18.3.1 EXTI configurable event input wake-up

The figure below is a detailed representation of the logic associated with configurable event inputs that wake up the CPU sub-system bus clocks and generate an EXTI pending flag and interrupt to the CPU, and/or a CPU wake-up event.

Figure 92. Configurable event trigger logic CPU wake-up



1. Only for the input events that support CPU rxev generation c_event.

The software interrupt event register allows configurable events to be triggered by software, writing the corresponding register bit, whatever the edge selection setting.

The configurable event active trigger edge (or both edges) is selected and enabled in the rising/falling edge selection registers.

The CPU has its dedicated wake-up (interrupt) mask register and a dedicated event mask registers. When the event is enabled, it is generated to the CPU. All events for the CPU are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request registers. The pending register is only set for an unmasked interrupt. Each configurable event provides a common interrupt to the CPU. The configurable event interrupts must be acknowledged by software in the EXTI_RPR and/or EXTI_FPR registers.

When a CPU wake-up (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

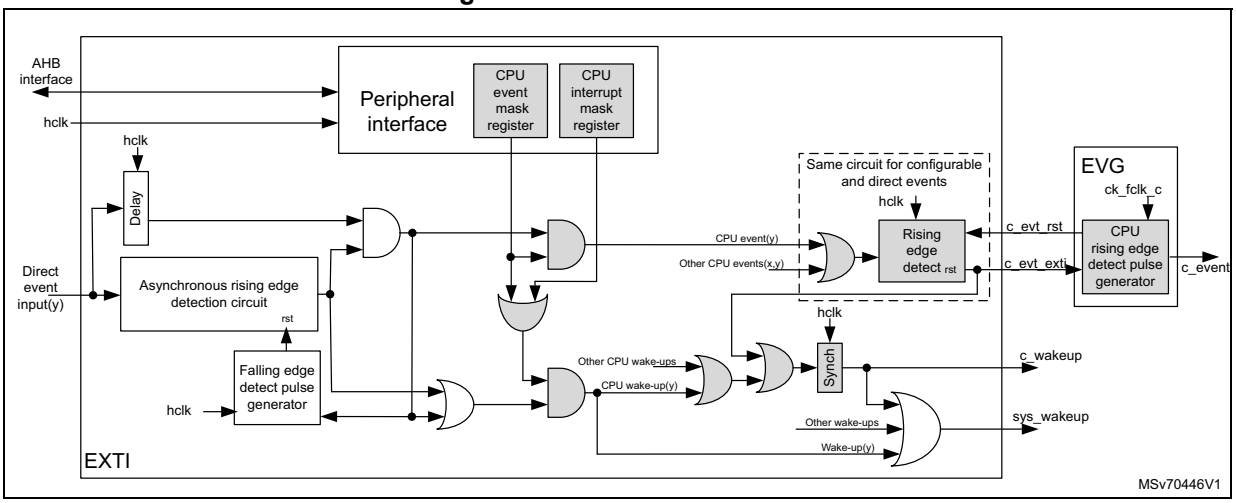
Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

18.3.2 EXTI direct event input wake-up

The direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU subsystem clocks, and can generate a CPU wake-up event. The peripheral synchronous interrupt, associated with the direct wake-up event, wakes up the CPU.

The EXTI direct event is able to generate a CPU event that wakes up the CPU. The CPU event may occur before the associated peripheral interrupt flag is set.

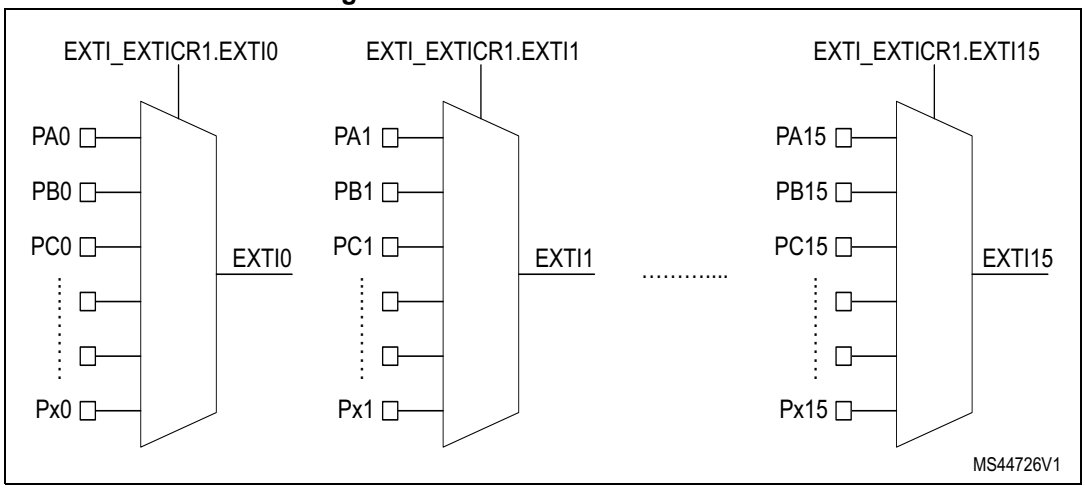
Figure 93. EXTI direct events



18.3.3 EXTI mux selection

The EXTI mux allows the selection of GPIOs as interrupts and wake-up. GPIOs are connected via 16 EXTI mux lines to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI mux output is controlled in the *EXTI external interrupt selection register (EXTI_EXTICR1)*.

Figure 94. EXTI mux GPIO selection



The EXTI mux outputs are available as output signals from the EXTI to trigger other peripherals, whatever the masking in EXTI_IMR and EXTI_EMR registers.

18.4 EXTI functional behavior

The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the CPU wake-up generation is conditioned by the CPU interrupt mask and CPU event mask.

Table 151. Masking functionality

CPU interrupt enable (in EXTI_IMR.IMn)	CPU event enable (in EXTI_EMR.EMn)	Configurable event inputs (in EXTI_RPR.RPIFn and EXTI_FPR.FPIFn)	Exti(n) interrupt(1)	CPU event	CPU wake-up
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes(2)
	1	Status latched	Yes	Yes	Yes

1. The single exti(n) interrupt goes to the CPU. If no interrupt is required for CPU(m), the exti(n) interrupt must be masked in the CPU NVIC.
2. Only if CPU interrupt is enabled in EXTI_IMR.IMn.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending bits EXTI_RPR.RPIFn and/or EXTI_FPR.FPIFn is/are set: the CPU sub-system is woken up and the CPU interrupt signal is activated. The EXTI_RPR.RPIFn and/or

EXTI_FPR.FPIFn pending bits must be cleared by software writing it to 1. This action clears the CPU interrupt.

For the configurable event inputs, an event request can be generated by software when writing a 1 in the software interrupt/event register EXTI_SWIER, allowing the generation of a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

18.5 EXTI event protection

The EXTI is able to protect event register bits from being modified by non-secure and unprivileged accesses. The protection is individually activated per input event via the register bits in EXTI_SECCFGR and EXTI_PRIVCFGR. At EXTI level, the protection consists in preventing the following unauthorized write access:

- Change the settings of the secure and/or privileged configurable events.
- Change the masking of the secure and/or privileged input events.
- Clear pending status of the secure and/or privileged input events.

Table 152. Register protection overview

Register name	Access type	Protection(1)(2)
EXTI_RTSR	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_FTSR	RW	
EXTI_SWIER	RW	
EXTI_RPR	RW	
EXTI_FPR	RW	
EXTI_SECCFGR	RW	Always secure. Privilege can be bit-wise enabled in EXTI_PRIVCFGR.
EXTI_PRIVCFGR	RW	Always privilege. Security can be bit-wise enabled in EXTI_SECCFGR.
EXTI_EXTICRn	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_LOCKR	RW	Always secure
EXTI_IM	RW	Security and privilege can be bit-wise enabled in EXTI_SECCFGR and EXTI_PRIVCFGR.
EXTI_EMR	RW	

1. Security is enabled with the individual input event (EXTI_SECCFGR register).
2. Privilege is enabled with the individual Input event (EXTI_PRIVCFGR register).

18.5.1 EXTI security protection

When security is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a secure access. A non-secure write access is discarded and a read returns 0.

When input events are non-secure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and non-secure access.

The security configuration in registers EXTI_SECCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

18.5.2 EXTI privilege protection

When privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privileged access. An unprivileged write access is discarded and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privileged access and unprivileged access.

The privileged configuration in registers EXTI_PRIVCFGR can be globally locked after reset by EXTI_LOCKR.LOCK.

18.6 EXTI registers

Table 153. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [57:32] configuration
0x060 - 0x06C	EXTI IO port mux selection
0x070	EXTI protection lock configuration
0x080 - 0x0BC	CPU input event configuration

All registers can be accessed with word (32-bit), half-word (16-bit) and byte (8-bit) access.

18.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RTx:** Rising trigger event configuration bit of configurable event input x[(1)] (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, RTx can be accessed only with secure access. Non-secure write to this bit x is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, RTx can be accessed only with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FTx:** Falling trigger event configuration bit of configurable event input x [1] (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, FTx can be accessed only with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, FTx can be accessed only with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.

0: Falling trigger disabled (for event and Interrupt) for input line

1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **SWIx:** Software interrupt on event x (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can be accessed only with secure access. Non-secure write to this SWI x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can be accessed only with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

18.6.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF16
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **RPIFx:** configurable event inputs x rising edge pending bit (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, RPIFx can be accessed only with secure access. Non-secure write to this RPIFx is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can be accessed only with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

18.6.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **FPIFx:** configurable event inputs x falling edge pending bit (x = 16 to 0)

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can be accessed only with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can be accessed only with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

18.6.6 EXTI security configuration register (EXTI_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security, a non-secure write access is ignored and causes the generation of an illegal access event. A non-secure read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SECx:** Security enable on event input x (x = 31 to 0)

When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (non-secure)
1: Event security enabled (secure)

18.6.7 EXTI privilege configuration register (EXTI_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for privilege capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIVx:** Security enable on event input x (x = 31 to 0)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and non-secure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Non-secure write to this PRIVx is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

18.6.8 EXTI rising trigger selection register 2 (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT53	Res.	Res.	RT50	Res.	Res.
										rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **RT53:** Rising trigger event configuration bit of configurable event input x[1]

When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RTx can be accessed only with secure access. Non-secure write to this bit x is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RTx can be accessed only with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 18 **RT50:** Rising trigger event configuration bit of configurable event input x[1]

When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RTx can be accessed only with secure access. Non-secure write to this bit x is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RTx can be accessed only with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bit 14 **RT46:** Rising trigger event configuration bit of configurable event input x[1]

When EXTI_SECCFGR.SECx is disabled, RTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RTx can be accessed only with secure access. Non-secure write to this bit x is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RTx can be accessed only with privileged access. Unprivileged write to this bit x is discarded, unprivileged read returns 0.
0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.9 EXTI falling trigger selection register 2 (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT53	Res.	Res.	FT50	Res.	Res.
										rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **FT53:** Falling trigger event configuration bit of configurable event input x [1]
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can be accessed only with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can be accessed only with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 18 **FT50:** Falling trigger event configuration bit of configurable event input x [1]
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can be accessed only with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can be accessed only with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

Bit 14 **FT46:** Falling trigger event configuration bit of configurable event input x [1]
When EXTI_SECCFGR.SECx is disabled, FTx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FTx can be accessed only with secure access. Non-secure write to this FTx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FTx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FTx can be accessed only with privileged access. Unprivileged write to this FTx is discarded, unprivileged read returns 0.
0: Falling trigger disabled (for event and Interrupt) for input line
1: Falling trigger enabled (for event and Interrupt) for input line.

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

18.6.10 EXTI software interrupt event register 2 (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI53	Res.	Res.	SWI50	Res.	Res.
										rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SWI46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **SWI53:** Software interrupt on event x

When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can be accessed only with secure access. Non-secure write to this SWI x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can be accessed only with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

Bit 18 **SWI50:** Software interrupt on event x

When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can be accessed only with secure access. Non-secure write to this SWI x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can be accessed only with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

Bit 14 **SWI46:** Software interrupt on event x

When EXTI_SECCFGR.SECx is disabled, SWIx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, SWIx can be accessed only with secure access. Non-secure write to this SWI x is discarded, non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, SWIx can be accessed with unprivileged and privileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, SWIx can be accessed only with privileged access. Unprivileged write to this SWIx is discarded, unprivileged read returns 0.

A software interrupt is generated independent from the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

0: Writing 0 has no effect.

1: Writing 1 triggers a rising edge event on event x. This bit is auto cleared by hardware.

18.6.11 EXTI rising edge pending register 2 (EXTI_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF53	Res.	Res.	RPIF50	Res.	Res.
										rw			rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RPIF46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **RPIF53:** configurable event inputs x rising edge pending bit

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can be accessed only with secure access. Non-secure write to this RPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can be accessed only with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 18 **RPIF50:** configurable event inputs x rising edge pending bit

When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can be accessed only with secure access. Non-secure write to this RPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can be accessed only with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 14 **RPIF46:** configurable event inputs x rising edge pending bit
When EXTI_SECCFGR.SECx is disabled, RPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, RPIFx can be accessed only with secure access. Non-secure write to this RPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, RPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, RPIFx can be accessed only with privileged access. Unprivileged write to this RPIFx is discarded, unprivileged read returns 0.
0: No rising edge trigger request occurred
1: Rising edge trigger request occurred
This bit is set when the rising edge event or an EXTI_SWIER software trigger arrives on the configurable event line. This bit is cleared by writing 1 to it.

18.6.12 EXTI falling edge pending register 2 (EXTI_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

Contains only register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF53	Res.	Res.	FPIF50	Res.	Res.
										rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FPIF46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:22, 20:19, 17:15, 13:0 Reserved, must be kept at reset value.

Bit 21 **FPIF53:** configurable event inputs x falling edge pending bit
When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can be accessed only with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can be accessed only with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 18 **FPIF50:** configurable event inputs x falling edge pending bit

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can be accessed only with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can be accessed only with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

Bit 14 **FPIF46:** configurable event inputs x falling edge pending bit

When EXTI_SECCFGR.SECx is disabled, FPIFx can be accessed with non-secure and secure access.
When EXTI_SECCFGR.SECx is enabled, FPIFx can be accessed only with secure access. Non-secure write to this FPIFx is discarded, non-secure read returns 0.
When EXTI_PRIVCFGR.PRIVx is disabled, FPIFx can be accessed with unprivileged and privileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, FPIFx can be accessed only with privileged access. Unprivileged write to this FPIFx is discarded, unprivileged read returns 0.
0: No falling edge trigger request occurred
1: Falling edge trigger request occurred
This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing 1 to it.

18.6.13 EXTI security configuration register 2 (EXTI_SECCFGR2)

Address offset: 0x034

Reset value: 0x0000 0000

This register provides write access security, a non-secure write access is ignored and causes the generation of an illegal access event. A non-secure read returns the register data.

Contains only register bits for privilege capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SEC57	SEC56	SEC55	SEC54	SEC53	SEC52	SEC51	SEC50	SEC49	SEC48
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC47	SEC46	SEC45	SEC44	SEC43	SEC42	SEC41	SEC40	SEC39	SEC38	SEC37	SEC36	SEC35	SEC34	SEC33	SEC32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:0 **SECx:** Security enable on event input x (x = 57 to 32)
When EXTI_PRIVCFGR.PRIVx is disabled, SECx can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR.PRIVx is enabled, SECx can only be written with privileged access. Unprivileged write to this SECx is discarded.
0: Event security disabled (non-secure)
1: Event security enabled (secure)

18.6.14 EXTI privilege configuration register 2 (EXTI_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged read returns the register data.

Contains only register bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PRIV57	PRIV56	PRIV55	PRIV54	PRIV53	PRIV52	PRIV51	PRIV50	PRIV49	PRIV48
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV47	PRIV46	PRIV45	PRIV44	PRIV43	PRIV42	PRIV41	PRIV40	PRIV39	PRIV38	PRIV37	PRIV36	PRIV35	PRIV34	PRIV33	PRIV32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:0 **PRIVx:** Security enable on event input x (x = 57 to 32)
When EXTI_SECCFGR.SECx is disabled, PRIVx can be accessed with secure and non-secure access.
When EXTI_SECCFGR.SECx is enabled, PRIVx can only be written with secure access. Non-secure write to this PRIVx is discarded.
0: Event privilege disabled (unprivileged)
1: Event privilege enabled (privileged)

18.6.15 EXTI external interrupt selection register (EXTI_EXTICR1)

Address offset: 0x060

(EXTI mux 0, 1, 2, 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection

These bits are written by software to select the source input for EXTI3 external interrupt.
When EXTI_SECCFGR1.SEC3 is disabled, EXTI3 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC3 is enabled, EXTI3 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV3 is disabled, EXTI3 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV3 is enabled, EXTI3 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA3 pin
0x01: PB3 pin
0x02: PC3 pin
0x03: PD3 pin
0x04: PE3 pin
0x05: PF3 pin
0x06: PG3 pin
0x07: PH3 pin
0x08: PI3 pin
Others: reserved

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.
When EXTI_SECCFGR1.SEC2 is disabled, EXTI2 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC2 is enabled, EXTI2 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV2 is disabled, EXTI2 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV2 is enabled, EXTI2 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA2 pin
0x01: PB2 pin
0x02: PC2 pin
0x03: PD2 pin
0x04: PE2 pin
0x05: PF2 pin
0x06: PG2 pin
0x07: PH2 pin
0x08: PI2 pin
Others: reserved

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.
When EXTI_SECCFGR1.SEC1 is disabled, EXTI1 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC1 is enabled, EXTI1 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV1 is disabled, EXTI1 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV1 is enabled, EXTI1 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA1 pin
0x01: PB1 pin
0x02: PC1 pin
0x03: PD1 pin
0x04: PE1 pin
0x05: PF1 pin
0x06: PG1 pin
0x07: PH1 pin
0x08: PI1 pin
Others: reserved

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection

These bits are written by software to select the source input for EXTI0 external interrupt.
When EXTI_SECCFGR1.SEC0 is disabled, EXTI0 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC0 is enabled, EXTI0 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV0 is disabled, EXTI0 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV0 is enabled, EXTI0 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA0 pin
0x01: PB0 pin
0x02: PC0 pin
0x03: PD0 pin
0x04: PE0 pin
0x05: PF0 pin
0x06: PG0 pin
0x07: PH0 pin
0x08: PI0 pin
Others: reserved

18.6.16 EXTI external interrupt selection register (EXTI_EXTICR2)

Address offset: 0x064

(EXTI mux 4, 5, 6, 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection

These bits are written by software to select the source input for EXTI7 external interrupt.
When EXTI_SECCFGR1.SEC7 is disabled, EXTI7 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC7 is enabled, EXTI7 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV7 is disabled, EXTI7 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV7 is enabled, EXTI7 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA7 pin
0x01: PB7 pin
0x02: PC7 pin
0x03: PD7 pin
0x04: PE7 pin
0x05: PF7 pin
0x06: PG7 pin
0x07: PH7 pin
0x08: PI7 pin
Others: reserved

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection

These bits are written by software to select the source input for EXTI6 external interrupt.
When EXTI_SECCFGR1.SEC6 is disabled, EXTI6 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC6 is enabled, EXTI6 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV6 is disabled, EXTI6 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV6 is enabled, EXTI6 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA6 pin
0x01: PB6 pin
0x02: PC6 pin
0x03: PD6 pin
0x04: PE6 pin
0x05: PF6 pin
0x06: PG6 pin
0x07: PH6 pin
0x08: PI6 pin
Others: reserved

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection

These bits are written by software to select the source input for EXTI5 external interrupt.
When EXTI_SECCFGR1.SEC5 is disabled, EXTI5 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC5 is enabled, EXTI5 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV5 is disabled, EXTI5 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV5 is enabled, EXTI5 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA5 pin
0x01: PB5 pin
0x02: PC5 pin
0x03: PD5 pin
0x04: PE5 pin
0x05: PF5 pin
0x06: PG5 pin
0x07: PH5 pin
0x08: PI5 pin
Others: reserved

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection

These bits are written by software to select the source input for EXTI4 external interrupt.
When EXTI_SECCFGR1.SEC4 is disabled, EXTI4 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC4 is enabled, EXTI4 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV4 is disabled, EXTI4 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV4 is enabled, EXTI4 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA4 pin
0x01: PB4 pin
0x02: PC4 pin
0x03: PD4 pin
0x04: PE4 pin
0x05: PF4 pin
0x06: PG4 pin
0x07: PH4 pin
0x08: PI4 pin
Others: reserved

18.6.17 EXTI external interrupt selection register (EXTI_EXTICR3)

Address offset: 0x068

(EXTI mux 8, 9, 10, 11)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection

These bits are written by software to select the source input for EXTI11 external interrupt.
When EXTI_SECCFGR1.SEC11 is disabled, EXTI11 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC11 is enabled, EXTI11 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV11 is disabled, EXTI11 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV11 is enabled, EXTI11 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA11 pin
0x01: PB11 pin
0x02: PC11 pin
0x03: PD11 pin
0x04: PE11 pin
0x05: PF11 pin
0x06: PG11 pin
0x07: PH11 pin
0x08: PI11 pin
Others: reserved

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection

These bits are written by software to select the source input for EXTI10 external interrupt.
When EXTI_SECCFGR1.SEC10 is disabled, EXTI10 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC10 is enabled, EXTI10 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV10 is disabled, EXTI10 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV10 is enabled, EXTI10 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA10 pin
0x01: PB10 pin
0x02: PC10 pin
0x03: PD10 pin
0x04: PE10 pin
0x05: PF10 pin
0x06: PG10 pin
0x07: PH10 pin
0x08: PI10 pin
Others: reserved

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection

These bits are written by software to select the source input for EXTI9 external interrupt.
When EXTI_SECCFGR1.SEC9 is disabled, EXTI9 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC9 is enabled, EXTI9 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV9 is disabled, EXTI9 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV9 is enabled, EXTI9 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA9 pin
0x01: PB9 pin
0x02: PC9 pin
0x03: PD9 pin
0x04: PE9 pin
0x05: PF9 pin
0x06: PG9 pin
0x07: PH9 pin
0x08: PI9 pin
Others: reserved

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection

These bits are written by software to select the source input for EXTI8 external interrupt.
When EXTI_SECCFGR1.SEC8 is disabled, EXTI8 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC8 is enabled, EXTI8 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV8 is disabled, EXTI8 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV8 is enabled, EXTI8 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA8 pin
0x01: PB8 pin
0x02: PC8 pin
0x03: PD8 pin
0x04: PE8 pin
0x05: PF8 pin
0x06: PG8 pin
0x07: PH8 pin
0x08: PI8 pin
Others: reserved

18.6.18 EXTI external interrupt selection register (EXTI_EXTICR4)

Address offset: 0x06C

(EXTI mux 12, 13, 14, 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection

These bits are written by software to select the source input for EXTI15 external interrupt.
When EXTI_SECCFGR1.SEC15 is disabled, EXTI15 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC15 is enabled, EXTI15 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV15 is disabled, EXTI15 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV15 is enabled, EXTI15 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
0x03: PD15 pin
0x04: PE15 pin
0x05: PF15 pin
0x06: PG15 pin
0x07: PH15 pin
Others: reserved

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection

These bits are written by software to select the source input for EXTI14 external interrupt.
When EXTI_SECCFGR1.SEC14 is disabled, EXTI14 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC14 is enabled, EXTI14 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV14 is disabled, EXTI14 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV14 is enabled, EXTI14 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA14 pin
0x01: PB14 pin
0x02: PC14 pin
0x03: PD14 pin
0x04: PE14 pin
0x05: PF14 pin
0x06: PG14 pin
0x07: PH14 pin
Others: reserved

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection

These bits are written by software to select the source input for EXTI13 external interrupt.
When EXTI_SECCFGR1.SEC13 is disabled, EXTI13 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC13 is enabled, EXTI13 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV13 is disabled, EXTI13 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV13 is enabled, EXTI13 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA13 pin
0x01: PB13 pin
0x02: PC13 pin
0x03: PD13 pin
0x04: PE13 pin
0x05: PF13 pin
0x06: PG13 pin
0x07: PH13 pin
Others: reserved

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection

These bits are written by software to select the source input for EXTI12 external interrupt.
When EXTI_SECCFGR1.SEC12 is disabled, EXTI12 can be accessed with non-secure and secure access.
When EXTI_SECCFGR1.SEC12 is enabled, EXTI12 can be accessed only with secure access. Non-secure write is discarded and non-secure read returns 0.
When EXTI_PRIVCFGR1.PRIV12 is disabled, EXTI12 can be accessed with privileged and unprivileged access.
When EXTI_PRIVCFGR1.PRIV12 is enabled, EXTI12 can be accessed only with privileged access. Unprivileged write to this bit is discarded.
0x00: PA12 pin
0x01: PB12 pin
0x02: PC12 pin
0x03: PD12 pin
0x04: PE12 pin
0x05: PF12 pin
0x06: PG12 pin
0x07: PH12 pin
Others: reserved

18.6.19 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register provides write access security: a non-secure write access is ignored, a read access returns zero data, and both generate an illegal access event.

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

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

Bit 0 **LOCK**: Global security and privilege configuration registers (EXTI_SECCFGR and EXTI_PRIVCFGR) lock

This bit is written once after reset.

0: Security and privilege configuration open, can be modified.

1: Security and privilege configuration locked, can no longer be modified.

18.6.20 EXTI CPU wake-up with interrupt mask register (EXTI_IMR1)

Address offset: 0x080

Reset value: 0xFFFE 0000

Contains register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IMx:** CPU wake-up with interrupt mask on event input x [1] (x = 31 to 0)

When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, IMx can be accessed only with secure access. Non-secure write to this bit is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can be accessed only with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with interrupt request from input event x is masked.

1: Wake-up with interrupt request from input event x is unmasked.

Bit IM31 is not available on all devices, refer to *Table 150*. If not present, consider this bit as reserved, and keep it at reset value.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

18.6.21 EXTI CPU wake-up with event mask register (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **EMx:** CPU wake-up with event generation mask on event input x (x = 31 to 0)

When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, EMx can be accessed only with secure access. Non-secure write to this bit x is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, EMx can be accessed only with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with event generation from Line x is masked.

1: Wake-up with event generation from Line x is unmasked.

Bit EM31 is not available on all devices, refer to *Table 150*. If not present, consider this bit as reserved, and keep it at reset value.

18.6.22 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x07DB BFFF

Contains register bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	IM58	IM57	IM56	IM55	IM54	IM53	IM52	IM51	IM50	IM49	IM48
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **IMx:** CPU wake-up with interrupt mask on event input x [1] (x = 58 to 0)

When EXTI_SECCFGR.SECx is disabled, IMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, IMx can be accessed only with secure access. Non-secure write to this bit is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, IMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, IMx can be accessed only with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with interrupt request from input event x is masked.

1: Wake-up with interrupt request from input event x is unmasked.

Bits IM[58:54], IM[46:44], and IM[36:32] are not available on all devices, refer to *Table 150*. If not present, consider these bits as reserved, and keep them at reset value.

1. The reset value for configurable event inputs is set to 0 in order to disable the interrupt by default.

18.6.23 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	EM58	EM57	EM56	EM55	EM54	EM53	EM52	EM51	EM50	EM49	EM48
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM47	EM46	EM45	EM44	EM43	EM42	EM41	EM40	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:0 **EMx:** CPU wake-up with event generation mask on event input x (x = 58 to 32)

When EXTI_SECCFGR.SECx is disabled, EMx can be accessed with non-secure and secure access.

When EXTI_SECCFGR.SECx is enabled, EMx can be accessed only with secure access. Non-secure write to this bit x is discarded and non-secure read returns 0.

When EXTI_PRIVCFGR.PRIVx is disabled, EMx can be accessed with privileged and unprivileged access.

When EXTI_PRIVCFGR.PRIVx is enabled, EMx can be accessed only with privileged access. Unprivileged write to this bit is discarded.

0: Wake-up with event generation from Line x is masked.

1: Wake-up with event generation from Line x is unmasked.

Bits EM[58:54], EM[46:44], and EM[36:32] are not available on all devices, refer to *Table 150*. If not present, consider these bits as reserved, and keep them at reset value.

18.6.24 EXTI register map

Table 154. EXTI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**EXTI_RTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT16	RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	**EXTI_FTSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT16	FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	**EXTI_SWIER1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI16	SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	**EXTI_RPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF1	RPIF1	RPIF1	RPIF1	RPIF1	RPIF1	RPIF1	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	**EXTI_FPR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF16	FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	**EXTI_ SECCFGR1**	SEC31	SEC30	SEC29	SEC28	SEC27	SEC26	SEC25	SEC24	SEC23	SEC22	SEC21	SEC20	SEC19	SEC18	SEC17	SEC16	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	**EXTI_ PRIVCFGR1**	PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	PRIV16	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	**EXTI_RTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT53	Res.	Res.	RT50	Res.	Res.	Res.	RT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0				0														
0x024	**EXTI_FTSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT53	Res.	Res.	FT50	Res.	Res.	Res.	FT46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0				0														
0x028	**EXTI_SWIER2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI53	Res.	Res.	SWI50	Res.	Res.	Res.	SWI46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0				0														
0x02C	**EXTI_RPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF53	Res.	Res.	RPIF50	Res.	Res.	Res.	RPIF46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0				0														
0x030	**EXTI_FPR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF53	Res.	Res.	FPIF50	Res.	Res.	Res.	FPIF46	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0			0				0														
0x034	**EXTI_ SECCFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	SEC57	SEC56	SEC55	SEC54	SEC53	SEC52	SEC51	SEC50	SEC49	SEC48	SEC47	SEC46	SEC45	SEC44	SEC43	SEC42	SEC41	SEC40	SEC39	SEC38	SEC37	SEC36	SEC35	SEC34	SEC33	SEC32
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**EXTI_ PRIVCFGR2**	Res.	Res.	Res.	Res.	Res.	Res.	PRIV57	PRIV56	PRIV55	PRIV54	PRIV53	PRIV52	PRIV51	PRIV50	PRIV49	PRIV48	PRIV47	PRIV46	PRIV45	PRIV44	PRIV43	PRIV42	PRIV41	PRIV40	PRIV39	PRIV38	PRIV37	PRIV36	PRIV35	PRIV34	PRIV33	PRIV32
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C to 0x05C	Reserved	Reserved																															

Table 154. EXTI register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060	**EXTI_EXTICR1**	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]								EXTI0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	**EXTI_EXTICR2**	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]								EXTI4[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	**EXTI_EXTICR3**	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]								EXTI8[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	**EXTI_EXTICR4**	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]								EXTI12[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	**EXTI_LOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK
	Reset value																																0
0x074 to 0x07C	Reserved	Reserved																															
0x080	**EXTI_IMR1**(1)	IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	**EXTI_EMR1**(1)	EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	EM16	EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	**EXTI_IMR2**(1)	Res.	Res.	Res.	Res.	Res.	IM58	IM57	IM56	IM55	IM54	IM53	IM52	IM51	IM50	IM49	IM48	IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
	Reset value						1	1	1	1	1	0	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x094	**EXTI_EMR2**(1)	Res.	Res.	Res.	Res.	Res.	EM58	EM57	EM56	EM55	EM54	EM53	EM52	EM51	EM50	EM49	EM48	EM47	EM46	EM45	EM44	EM43	EM42	EM41	EM40	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
	Reset value						1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. Some bits in this register are not available on some devices, refer to *Table 150*. If not present, consider these bits as reserved, and keep them at reset value.

Refer to *Section 2.3* for the register boundary addresses.

19 Cyclic redundancy check calculation unit (CRC)

19.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

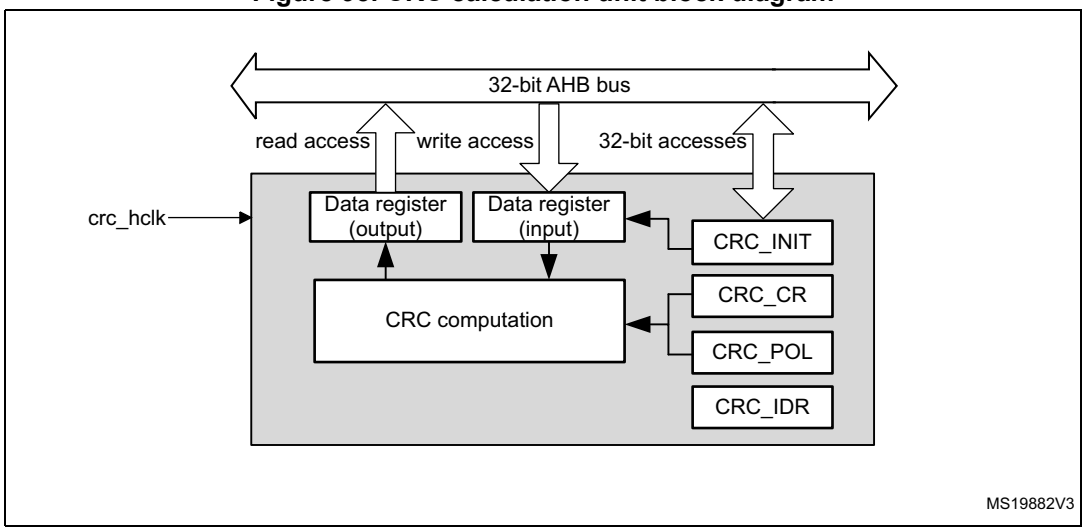
19.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

19.3 CRC functional description

19.3.1 CRC block diagram

Figure 95. CRC calculation unit block diagram



19.3.2 CRC internal signals

Table 155. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

19.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

19.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

19.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

19.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

19.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data.

0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

This bitfield controls the reversal of the bit order of the input data

00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word

11: Bit reversal done by word

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

19.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value
This register is used to write the CRC initial value.

19.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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

Bits 31:0 **POL[31:0]**: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

19.4.6 CRC register map

Table 156. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

20 CORDIC coprocessor (CORDIC)

This section applies only to STM32H562/63/73xx devices.

20.1 CORDIC introduction

The CORDIC coprocessor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing, and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making possible the use of a lower operating frequency, or freeing up processor cycles to perform other tasks.

20.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atan2, atanh, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

20.3 CORDIC functional description

20.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle θ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle θ. The x and y cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of θ. Inversely, the angle of a vector [x, y] corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

20.3.2 CORDIC functions

The first step when using the coprocessor is to select the required function, by programming the FUNC field of the CORDIC_CR register.

Table 157 lists the functions supported by the CORDIC coprocessor.

Table 157. CORDIC functions

Function	Primary argument (ARG1)	Secondary argument (ARG2)	Primary result (RES1)	Secondary result (RES2)
Cosine	angle θ	modulus m	$m \cdot \cos\theta$	$m \cdot \sin\theta$
Sine	angle θ	modulus m	$m \cdot \sin\theta$	$m \cdot \cos\theta$
Phase	x	y	atan2(y,x)	$\sqrt{x^2+y^2}$
Modulus	x	y	$\sqrt{x^2+y^2}$	atan2(y,x)
Arctangent	x	none	$\tan^{-1} x$	none
Hyperbolic cosine	x	none	cosh x	sinh x
Hyperbolic sine	x	none	sinh x	cosh x
Hyperbolic arctangent	x	none	$\tanh^{-1} x$	none
Natural logarithm	x	none	ln x	none
Square root	x	none	$\sqrt{x}$	none

Several functions take two input arguments (ARG1 and ARG2), and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: sin θ also generates cos θ, and cos θ also generates sin θ. Similarly for rectangular-to-polar conversion (phase(x,y), modulus(x,y)) and for hyperbolic functions (cosh θ, sinh θ).

Note: The exponential function, exp x, can be obtained as the sum of sinh x and cosh x. Furthermore, base N logarithms, $\log_N x$, can be derived by multiplying ln x by a constant K, where K = 1/ln N.

For certain functions (atan, log, sqrt) a scaling factor (see *Section 20.3.4*) can be applied to extend the range beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

Cosine

Table 158. Cosine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m · cos θ	[-1, 1]
RES2	m · sin θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the cosine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

Sine

Table 159. Sine parameters

Parameter	Description	Range
ARG1	Angle θ in radians, divided by π	[-1, 1]
ARG2	Modulus m	[0, 1]
RES1	m · sin θ	[-1, 1]
RES2	m · cos θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the sine of an angle in the range -π to π. It can also be used to perform polar to rectangular conversion.

The primary argument is the angle θ in radians. It must be divided by π before programming ARG1.

The secondary argument is the modulus m. If m is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

Phase

Table 160. Phase parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Phase angle θ in radians, divided by π	[-1, 1]
RES2	Modulus m	[0, 1]
SCALE	Not applicable	0

This function calculates the phase angle in the range -π to π of a vector **v** = [x y] (also known as atan2(y,x). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle θ of the vector **v**. RES1 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If |**v**| > 1 the result in RES2 is saturated to 1.

Modulus

Table 161. Modulus parameters

Parameter	Description	Range
ARG1	x coordinate	[-1, 1]
ARG2	y coordinate	[-1, 1]
RES1	Modulus m	[0, 1]
RES2	Phase angle θ	[-1, 1]
SCALE	Not applicable	0

This function calculates the magnitude, or modulus, of a vector **v** = [x y]. It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If |x| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If |y| > 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: $|v| = \sqrt{x^2 + y^2}$. If $|\mathbf{v}| > 1$ the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle θ of the vector **v**. RES2 must be multiplied by π to obtain the angle in radians. Note that values close to π may sometimes wrap to -π due to the circular nature of the phase angle.

Arctangent

Table 162. Arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-1, 1]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \tan^{-1} x$, in radians, divided by p	[-1, 1]
RES2	Not applicable	-
SCALE	n	[0 7]

This function calculates the arctangent, or inverse tangent, of the input argument x.

The primary argument, ARG1, is the input value, x = tan θ. If |x| > 1, a scaling factor of 2^{-n} must be applied in software such that $-1 < x \cdot 2^{-n} < 1$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the scale factor n must be programmed in the SCALE parameter.

Note that the maximum input value allowed is tan θ = 128, which corresponds to an angle θ = 89.55 degrees. For |x| > 128, a software method must be used to find $\tan^{-1} x$.

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle $\theta = \tan^{-1} x$. RES1 must be multiplied by $2^n \cdot \pi$ to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 163. Hyperbolic cosine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \cosh x$	[0.5 0.846]
RES2	$2^{-n} \cdot \sinh x$	[-0.683 0.683]
SCALE	n	1

This function calculates the hyperbolic cosine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. Since the minimum value of cosh x is 1, which is beyond the range of the q1.31 format, a scaling factor of 2^{-n} must be applied in software. The factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

Table 164. Hyperbolic sine parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.559, 0.559]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \sinh x$	[-0.683, 0.683]
RES2	$2^{-n} \cdot \cosh x$	[0.5, 0.846]
SCALE	n	1

This function calculates the hyperbolic sine of a hyperbolic angle x. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle x. Only values of x in the range -1.118 to +1.118 are supported. For all input values, a scaling factor of 2^{-n} must be applied in software, where n = 1. The scaled value $x \cdot 0.5$ is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

Table 165. Hyperbolic arctangent parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[-0.403 0.403]
ARG2	Not applicable	-
RES1	$2^{-n} \cdot \text{atanh } x$	[-0.559 0.559]

Table 165. Hyperbolic arctangent parameters (continued)

Parameter	Description	Range
RES2	Not applicable	-
SCALE	n	1

This function calculates the hyperbolic arctangent of the input argument x.

The primary argument is the input value x. Only values of x in the -0.806 to +0.806 range are supported. The value x must be scaled by a factor 2^{-n}, where n = 1. The scaled value x · 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, atanh x. RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

Natural logarithm

Table 166. Natural logarithm parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.054 0.875]
ARG2	Not applicable	-
RES1	$2^{-(n+1)}.\ln x$	[-0.279 0.137]
RES2	Not applicable	-
SCALE	n	[1 4]

This function calculates the natural logarithm of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.107 to 9.35 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < 1-2^{-n}$. The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 167 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 167. Natural log scaling factors and corresponding ranges

n	x range	ARG1 range
1	$0.107 \le x < 1$	$0.0535 \le ARG1 < 0.5$
2	$1 \le x < 3$	$0.25 \le ARG1 < 0.75$
3	$3 \le x < 7$	$0.375 \le ARG1 < 0.875$
4	$7 \le x \le 9.35$	$0.4375 \le ARG1 < 0.584$

The secondary argument is not used.

The primary result is the natural logarithm, ln x. RES1 must be multiplied by $2^{(n+1)}$ to obtain the correct value.

The secondary result is not used.

Square root

Table 168. Square root parameters

Parameter	Description	Range
ARG1	$x \cdot 2^{-n}$	[0.027 0.875]
ARG2	Not applicable	-
RES1	$2^{-n}\sqrt{x}$	[0.04 1]
RES2	Not applicable	-
SCALE	n	[0 2]

This function calculates the square root of the input argument x.

The primary argument is the input value x. Only values of x in the range 0.027 to 2.34 are supported. The value x must be scaled by a factor 2^{-n}, such that $x \cdot 2^{-n} < (1 - 2^{(-n-2)})$.

The scaled value $x \cdot 2^{-n}$ is programmed in ARG1 and the factor n must be programmed in the SCALE parameter.

Table 169 lists the valid scaling factors, n, and the corresponding ranges of x and ARG1.

Table 169. Square root scaling factors and corresponding ranges

n	x range	ARG1 range
0	$0.027 \leq x < 0.75$	$0.027 \leq ARG1 < 0.75$
1	$0.75 \leq x < 1.75$	$0.375 \leq ARG1 < 0.875$
2	$1.75 \leq x \leq 2.341$	$0.4375 \leq ARG1 \leq 0.585$

The secondary argument is not used.

The primary result is the square root of x. RES1 must be multiplied by 2^n to obtain the correct value.

The secondary result is not used.

20.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to $1 - 2^{-31}$ (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to $1 - 2^{-15}$ (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

20.3.4 Scaling factor

Several of the functions listed in *Section 20.3.2* specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output, or internal registers. If the scaling factor is required, it must be calculated by software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

20.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see *Figure 96*.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see *Figure 97*). The square root function converges at roughly twice the speed of the hyperbolic functions (see *Figure 98*).

Figure 96. CORDIC convergence for trigonometric functions

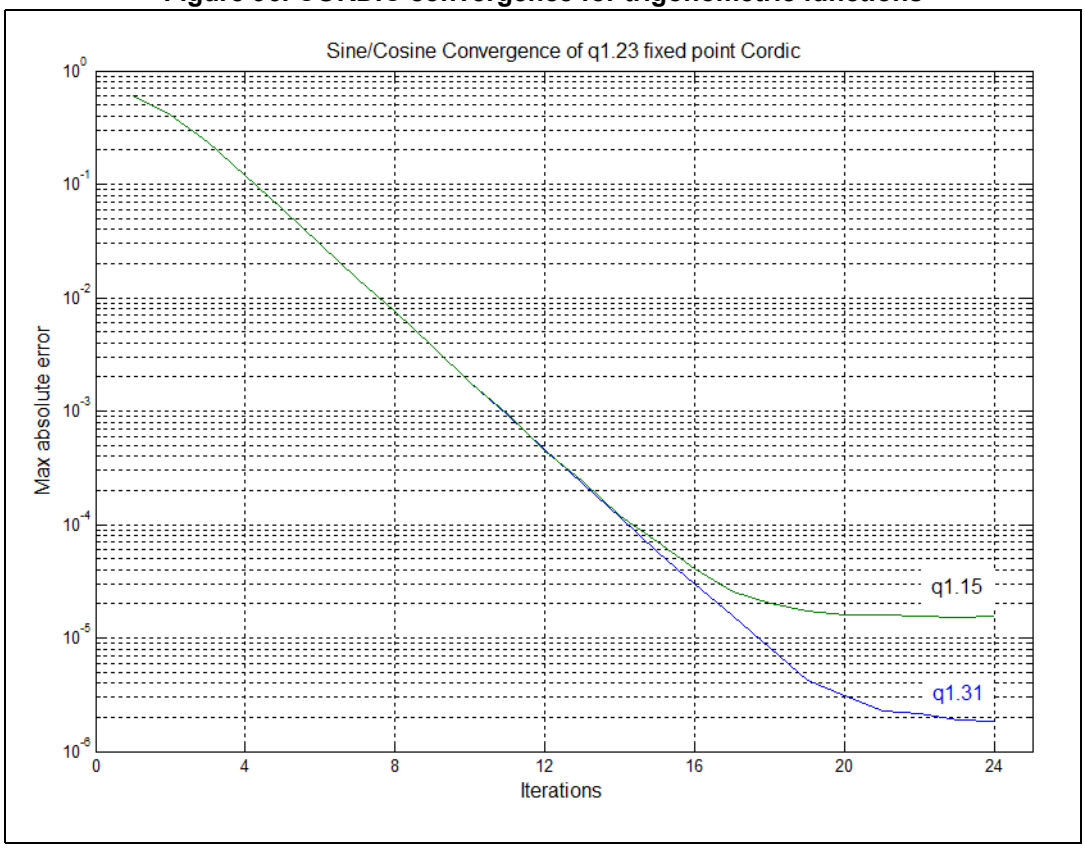


Figure 97. CORDIC convergence for hyperbolic functions

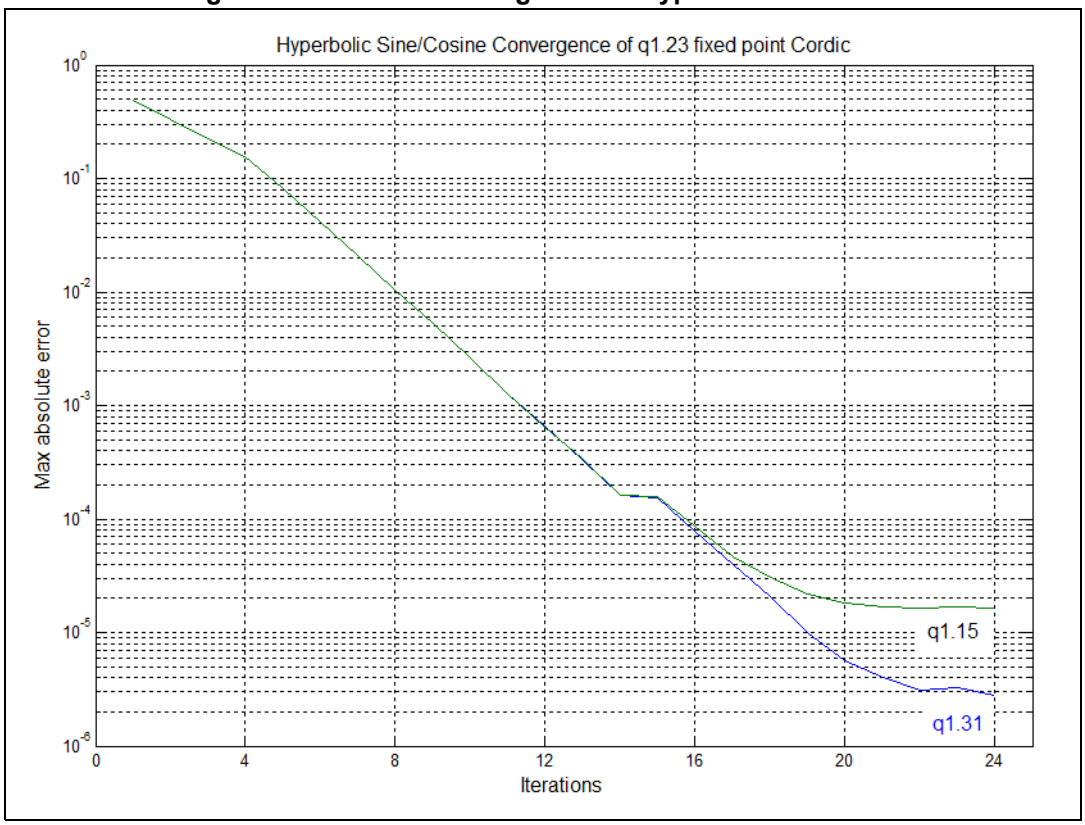
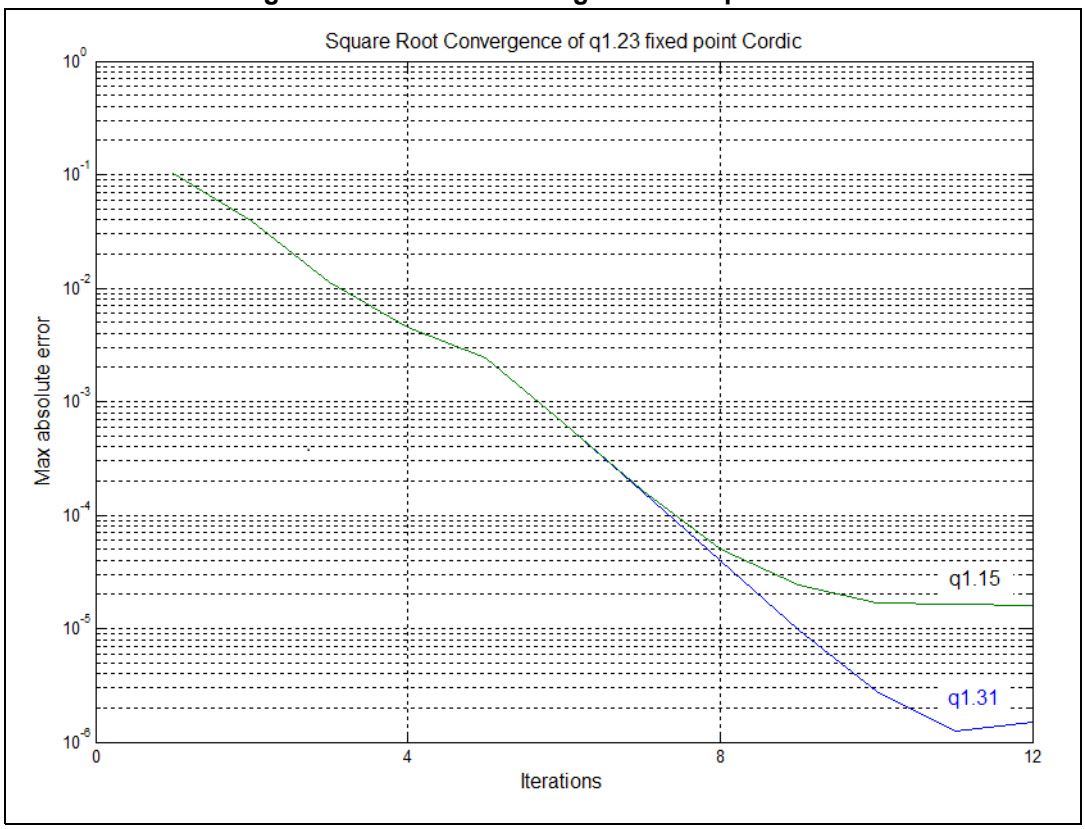


Figure 98. CORDIC convergence for square root



Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in *Table 170*, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The coprocessor stops as soon as the programmed number of iterations is completed, and the result can be read immediately.

Table 170. Precision vs. number of iterations

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sin, Cos, Phase(2), Mod, Atan(4)	4	1	2^{-3}	2^{-3}
	8	2	2^{-7}	2^{-7}
	12	3	2^{-11}	2^{-11}
	16	4	2^{-15}	2^{-15}
	20	5	2^{-18}	2^{-16}
	24	6	2^{-19}	2^{-16}

Table 170. Precision vs. number of iterations (continued)

Function	Number of iterations	Number of cycles	Max residual error(1)	
			q1.31 format	q1.15 format
Sinh, Cosh, Atanh, Ln(3)	4	1	2^{-2}	2^{-2}
	8	2	2^{-6}	2^{-6}
	12	3	2^{-10}	2^{-10}
	16	4	2^{-13}	2^{-13}
	20	5	2^{-17}	2^{-15}
	24	6	2^{-18}	2^{-15}
Sqrt(4)	4	1	2^{-7}	2^{-7}
	8	2	2^{-14}	2^{-14}
	12	3	2^{-19}	2^{-15}

1. Max residual error is the maximum error remaining after the given number of iterations, compared to the identical calculation performed in double precision floating point. An additional rounding error may be incurred, of up to 2^{-16} for q15 format or 2^{-20} for q31 format.
2. For modulus > 0.5. The achievable precision reduces proportionally to the magnitude of the modulus, as quantization error becomes significant.
3. SCALE = 1. If a higher scaling factor is used, the achievable precision is reduced proportionally.
4. SCALE = 0. If a higher scaling factor is used, the achievable precision is reduced proportionally.

20.3.6 Zero-overhead mode

The fastest way to use the coprocessor is to preprogram the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

The calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished), a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to reprogram the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument (ARG1) must be written first, followed by the secondary argument (ARG2). If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.

Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait-states until the calculation is completed, before returning the result. It is then possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation, and prepare the next input data, if needed. The CORDIC_CSR register can be reprogrammed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous ones have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. IF NRES = 1, a second read of CORDIC_RDATA must be performed to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be a calculation in progress, or waiting for the results to be read, and an operation pending. Any further access to CORDIC_WDATA while an operation is pending cancels it and overwrites the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:

1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step *3*.

20.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice, depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. The processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (for example, if low latency interrupts must be serviced).

20.3.8 Interrupt mode

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However, it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

20.3.9 DMA mode

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention. This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. The sequence is started by the processor setting the DMAWEN flag, the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generates a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

Note: No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore useful only when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.

Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and may block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

20.4 CORDIC registers

The CORDIC registers can be accessed only in 32-bit word format

20.4.1 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARG SIZE	RES SIZE	N ARGS	NRES	DMA WEN	DMA REN	IEN
r									rw	rw	rw	rw	rw	rw	rw

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

Bit 31 **RRDY**: Result ready flag

0: No new data in output register

1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data

0: 32-bit

1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format. The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data

0: 32-bit

1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.

Bit 20 **NARGS**: Number of arguments expected by the CORDIC_WDATA register

0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.

1: Two 32-bit values must be written to the CORDIC_WDATA register to trigger the next calculation.

Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC_RDATA register

0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC_RDATA register on completion of the next calculation. One read from CORDIC_RDATA resets the RRDY flag.

1: Two 32-bit values are transferred to the CORDIC_RDATA register on completion of the next calculation. Two reads from CORDIC_RDATA are necessary to reset the RRDY flag.

Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel

0: Disabled. No DMA write requests are generated.

1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel

0: Disabled. No DMA read requests are generated.

1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt.

0: Disabled. No interrupt requests are generated.

1: Enabled. An interrupt request is generated whenever the RRDY flag is set.

This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor

The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor 2^{-n}, and/or the results need to be multiplied by 2^{n}. Refer to *Section 20.3.2* for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)

0: reserved

1 to 15: (Number of iterations)/4

To determine the number of iterations needed for a given accuracy refer to *Table 170*.

Note that for most functions, the recommended range for this field is 3 to 6.

Bits 3:0 **FUNC[3:0]**: Function

0: Cosine
1: Sine
2: Phase
3: Modulus
4: Arctangent
5: Hyperbolic cosine
6: Hyperbolic sine
7: Arctanh
8: Natural logarithm
9: Square root
10 to 15: Reserved

20.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

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

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to *Section 20.3.2* for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG 2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrite the argument data.

20.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:0 **RES[31:0]**: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed.

A read from this register resets the RRDY flag in the CORDIC_CSR register.

20.4.4 CORDIC register map

Table 171. CORDIC register map and reset value

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CORDIC_CSR	RRDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARGSIZE	RESSIZE	NARGS	NRES	DMAWEN	DMAREN	IEN	Res.	Res.	Res.	Res.	SCALE [2:0]				PRECISION [3:0]				FUNC [3:0]			
	Reset value	0									0	0	0	0	0	0	0					0	0	0	0	0	1	0	1	0	0	0	0
0x04	CORDIC_WDATA	ARG[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x08	CORDIC_RDATA	RES[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

21 Filter math accelerator (FMAC)

The FMAC section only applies to STM32H56x/573 MCUs.

21.1 FMAC introduction

The filter math accelerator unit performs arithmetic operations on vectors. It comprises a multiplier/accumulator (MAC) unit, together with address generation logic which allows it to index vector elements held in local memory.

The unit includes support for circular buffers on input and output, which allows digital filters to be implemented. Both finite and infinite impulse response filters can be realized.

The unit allows frequent or lengthy filtering operations to be offloaded from the CPU, freeing up the processor for other tasks. In many cases it can accelerate such calculations compared to a software implementation, resulting in a speed-up of time critical tasks.

21.2 FMAC main features

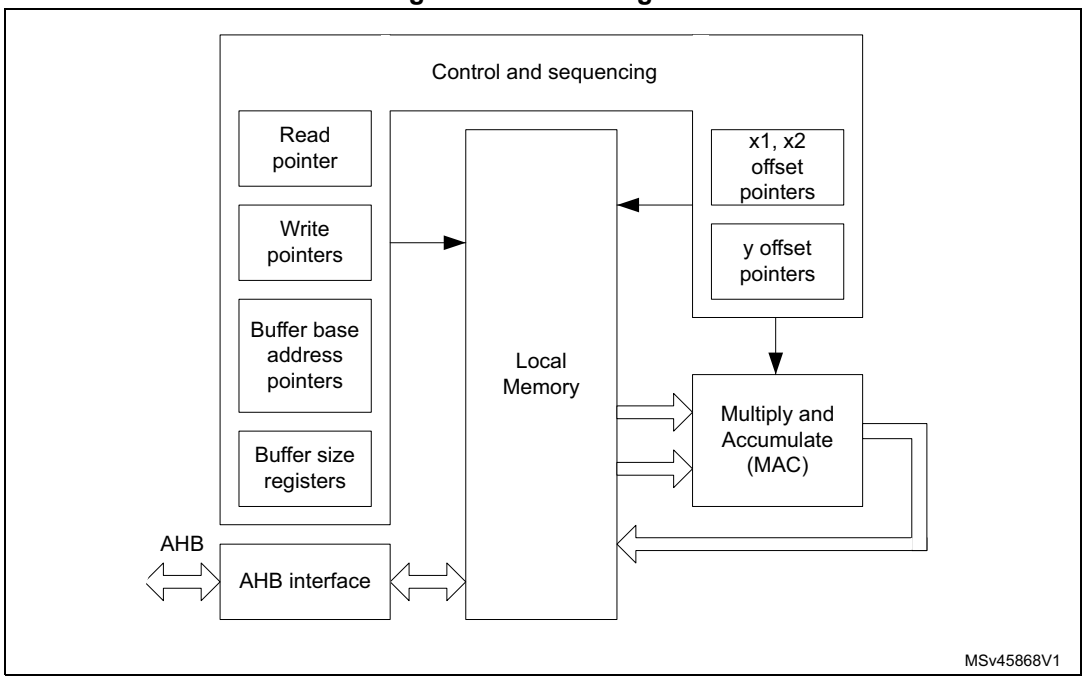
- 16 x 16-bit multiplier
- 24 + 2-bit accumulator with addition and subtraction
- 16-bit input and output data
- 256 x 16-bit local memory
- Up to three areas can be defined in memory for data buffers (two input, one output), defined by programmable base address pointers and associated size registers
- Input and output buffers can be circular
- Filter functions: FIR, IIR (direct form 1)
- Vector functions: Dot product, convolution, correlation
- AHB slave interface
- DMA read and write data channels

21.3 FMAC functional description

21.3.1 General description

The FMAC is shown in *Figure 99*.

Figure 99. Block diagram



The unit is built around a fixed point multiplier and accumulator (MAC). The MAC can take two 16-bit input signed values from memory, multiply them together and add them to the contents of the accumulator. The address of the input values in memory is determined using a set of pointers. These pointers can be loaded, incremented, decremented or reset by the internal hardware. The pointer and MAC operations are controlled by a built-in sequencer in order to execute the requested operation.

To calculate a dot product, the two input vectors are loaded into the local memory by the processor or DMA controller, and the requested operation is selected and started. Each pair of input vector elements is fetched from memory, multiplied together and accumulated. When all the vector elements have been processed, the contents of the accumulator are stored in the local memory, from where they can be read out by the processor or DMA.

The finite impulse response (FIR) filter operation (also known as convolution) consists in repeatedly calculating the dot product of the coefficient vector and a vector of input samples, the latter being shifted by one sample delay, with the least recent sample being discarded and a new sample added, at each repetition.

The infinite impulse response (IIR) filter operation is the convolution of the feedback coefficients with the previous output samples, added to the result of the FIR convolution.

A more detailed description of the filter operations is given in *Section 21.3.6: Filter functions*.

21.3.2 Local memory and buffers

The unit contains a 256 x 16-bit read/write memory which is used for local storage:

- Input values (the elements of the input vectors) are stored in two buffers, X1 and X2.
- Output values (the results of the operations) are stored in another buffer, Y.
- The locations and sizes of the buffers are designated as follows:
  - x1_base: the base address of the X1 buffer
  - x2_base: the base address of the X2 buffer
  - y_base: the base address of the Y buffer
  - x1_buf_size: the number of 16-bit addresses allocated to the X1 buffer
  - x2_buf_size: the number of 16-bit addresses allocated to the X2 buffer
  - y_buf_size: the number of 16-bit addresses allocated to the Y buffer.

These parameters are programmed in the corresponding registers when configuring the unit.

The CPU (or DMA controller) can initialize the contents of each buffer using the Initialization functions (*Section 21.3.5: Initialization functions*) and writing to the write data register. The data is transferred to the location within the target buffer indicated by a write pointer. After each new write, the write pointer is incremented. When the write pointer reaches the end of the allocated buffer space, it wraps back to the base address. This feature is used to load the elements of a vector prior to an operation, or to initialize a filter and load filter coefficients.

Buffer configuration

The buffer sizes and base address offsets must be configured in the X1, X2 and Y buffer configuration registers. For each function, the required buffer size is specified in the function description in *Section 21.3.6: Filter functions*. The base addresses can be chosen anywhere in internal memory, provided that all buffers fit within the internal memory address range (0x00 to 0xFF), that is, base address + buffer size must be less than 256.

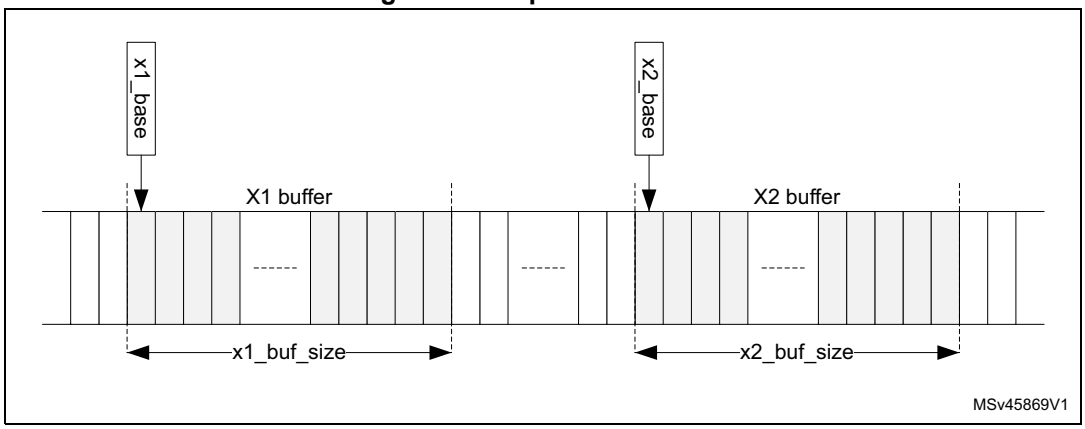
There is no constraint on the size and location of the buffers (they can overlap or even coincide exactly). For filter functions it is recommended not to overlap buffers as this can lead to erroneous behavior.

When circular buffer operation is required, an optional “headroom”, d, can be added to the buffer size. Furthermore, a watermark level can be set, to regulate the CPU or DMA activity. The value of d and the watermark level must be chosen according to the application performance requirements. For maximum throughput, the input buffer must never go empty, so d must be somewhat greater than the watermark level, allowing for any interrupt or DMA latency. On the other hand, if the input data can not be provided as fast as the unit can process them, the buffer can be allowed to empty waiting for the next data to be written, so d can be equal to the watermark level (to ensure that no overflow occurs on the input).

21.3.3 Input buffers

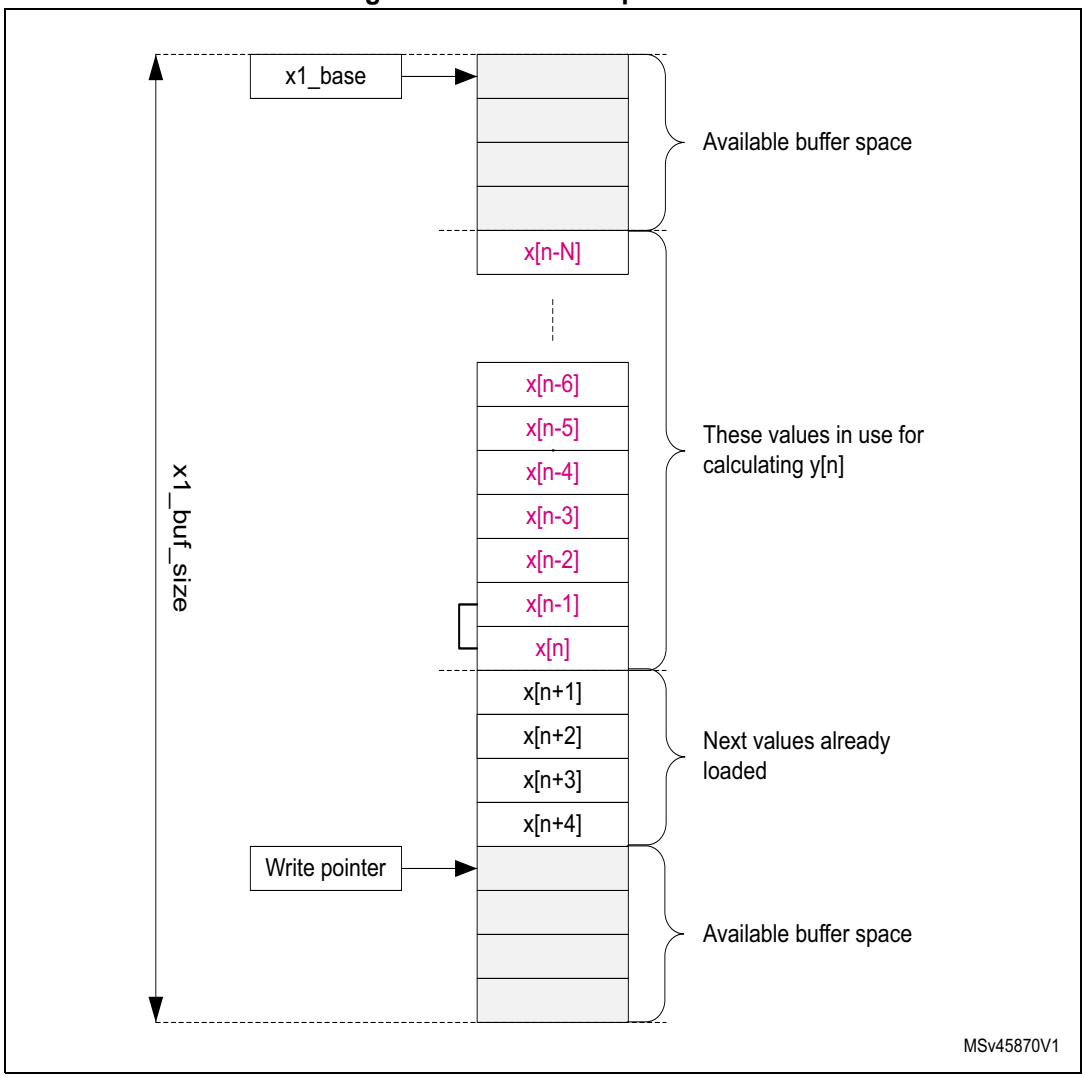
The X1 and X2 buffers are used to store data for input to the MAC. Each multiplication takes a value from the X1 buffer and a value from the X2 buffer and multiplies them together. A pointer in the control unit generates the read address offset (relative to the buffer base address) for each value. The pointers are managed by hardware according to the current function.

Figure 100. Input buffer areas



The X1 buffer can be used as a circular buffer, in which case new data are continually transferred into the input buffer whenever space is available. Pre-loading this buffer is optional for digital filters, since if no input samples have been written in the buffer when the operation is started, it is flagged as empty, which triggers the CPU or DMA to load new samples until there are enough to begin operation. Pre-loading is nevertheless useful in the case of a vector operation, that is, the input data is already available in system memory and circular operation is not required.

Figure 101. Circular input buffer



The X2 buffer can only be used in vector mode (that is not circular), and needs to be pre-loaded, except if the contents of the buffer do not change from one operation to the next. For filter functions, the X2 buffer is used to store the filter coefficients.

When operating as a circular buffer, the space allocated to the buffer (x1_buf_size) must generally be bigger than the number of elements in use for the current calculation, so that there are always new values available in the buffer. *Figure 101* illustrates the layout of the buffer for a filter operation. While calculating an output sample y[n], the unit uses a set of N+1 input samples, x[n-N] to x[n]. When this is finished, the unit starts the calculation of y[n+1], using the set of input samples x[n-N+1] to x[n+1]. The least-recent input sample, x[n-N], drops out of the input set, and a new sample, x[n+1], is added to it.

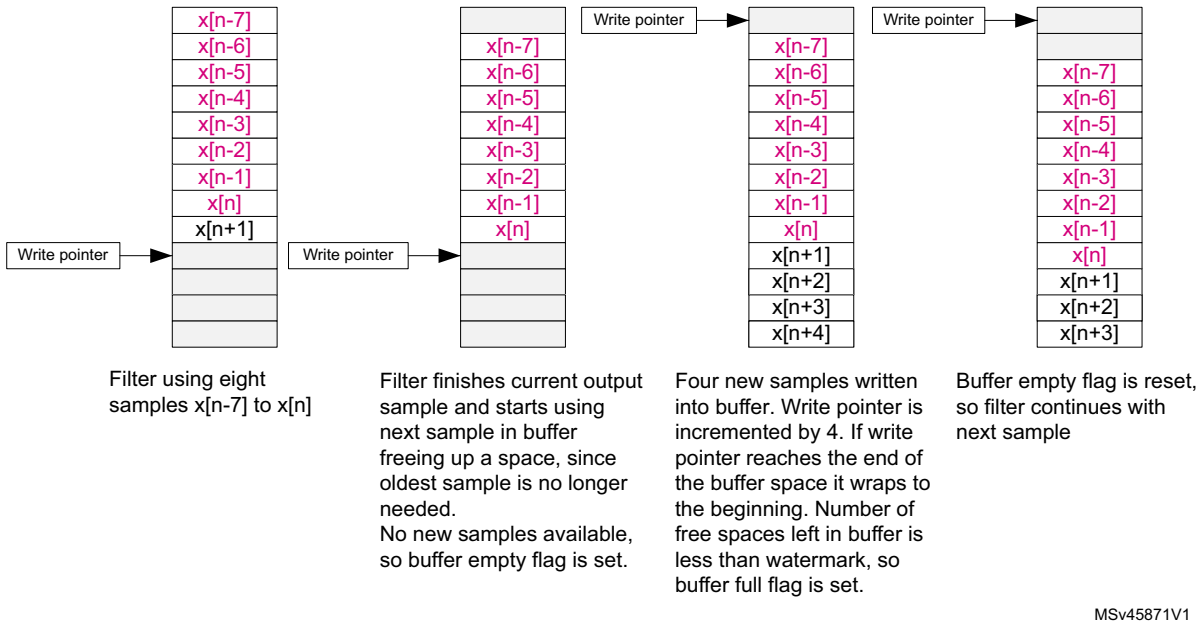
The processor, or DMA controller, must ensure that the new sample x[n+1] is available in the buffer space when required. If not, the buffer is flagged as empty, which stalls the execution of the unit until a new sample is added. No underflow condition is signaled on the X1 buffer.

Note: If the flow of samples is controlled by a timer or other peripheral such as an ADC, the buffer regularly goes empty, since the filter processes each new sample faster than the source can provide it. This is an essential feature of filter operation.

If the number of free spaces in the buffer is less than the watermark threshold programmed in the FULL_WM bitfield of the FMAC_X1BUFCFG register, the buffer is flagged as full. As long as the full flag is not set, interrupts are generated, if enabled, to request more data for the buffer. The watermark allows several data to be transferred under one interrupt, without danger of overflow. Nevertheless, if an overflow does occur, the OVFL error flag is set and the write data is ignored. The write pointer is not incremented in the event of an overflow.

The operation of the X1 buffer during a filtering operation is illustrated in *Figure 102*. This example shows an 8-tap FIR filter with a watermark set to four.

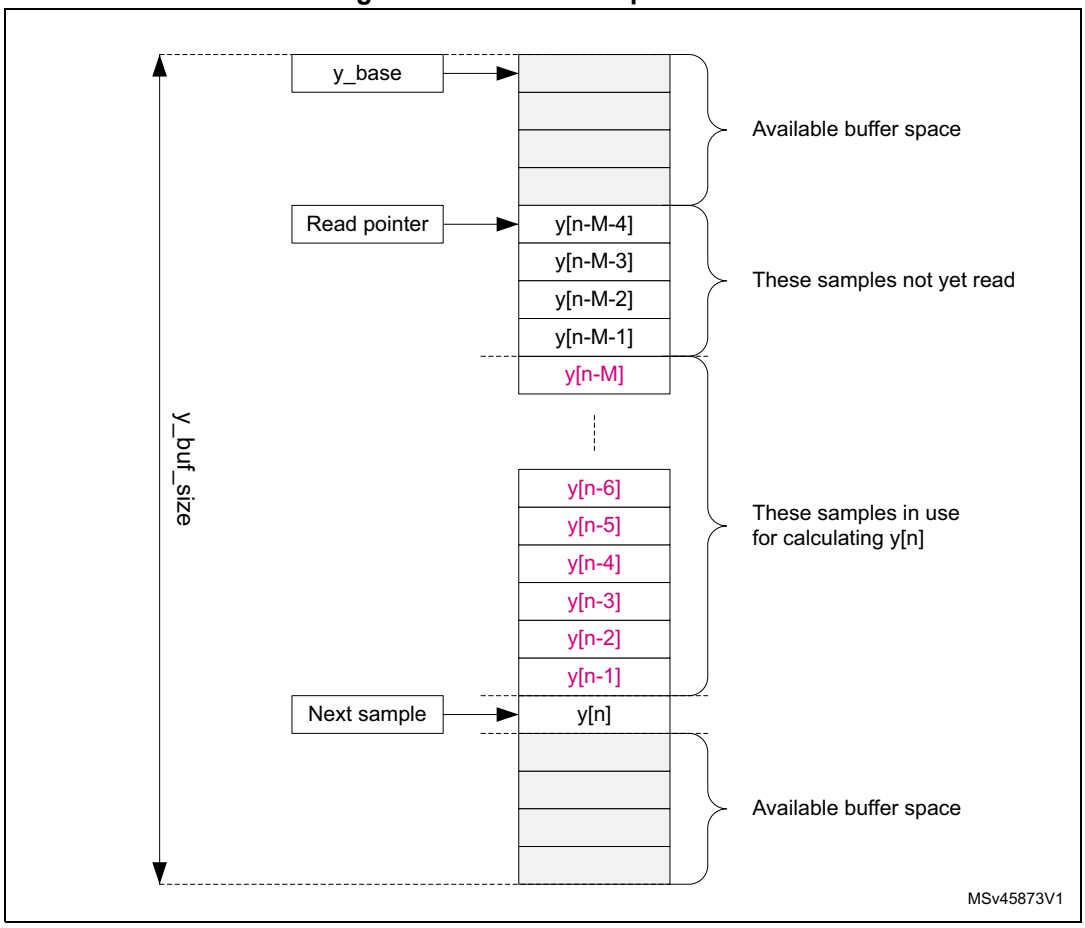
Figure 102. Circular input buffer operation



21.3.4 Output buffer

The Y (output) buffer is used to store the output of an accumulation. Each new output value is stored in the buffer until it is read by the processor or DMA controller. Each time a read access is made to the read data register, the read data is fetched from the address indicated by the read pointer. This pointer is incremented after each read, and wraps back to the base address when it reaches the end of the allocated Y buffer space.

Figure 103. Circular output buffer



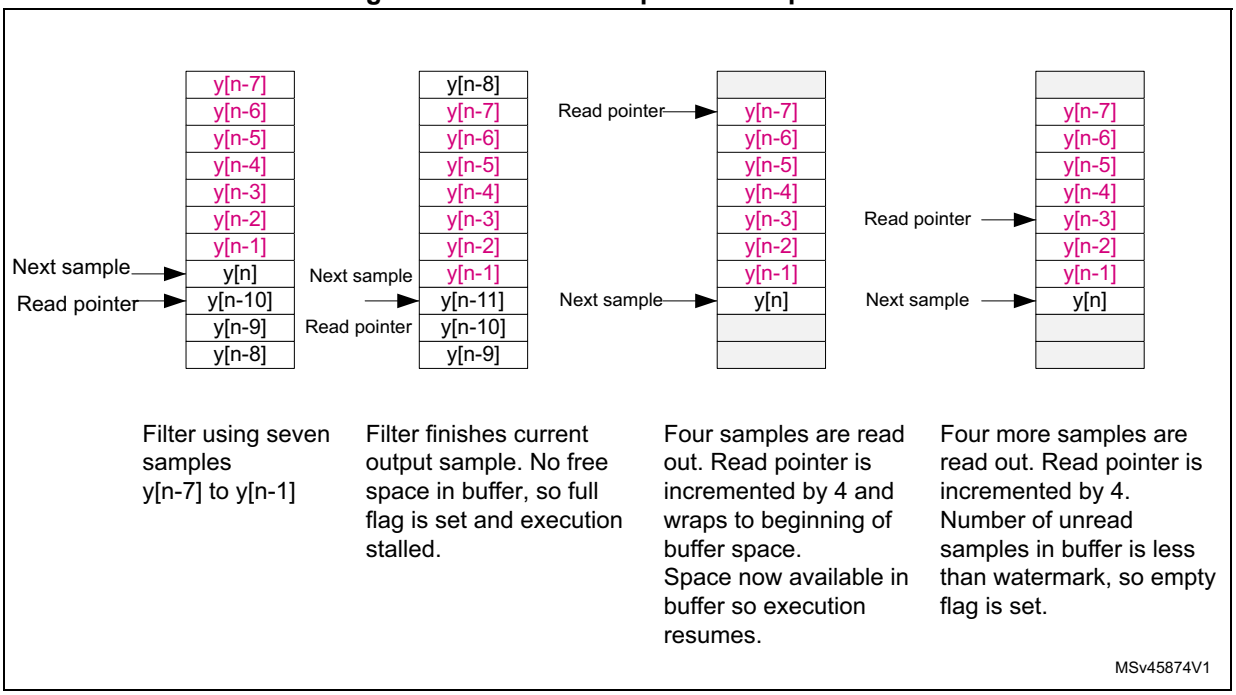
The Y buffer can also operate as a circular buffer. If the address for the next output value is the same as that indicated by the read pointer (an unread sample), then the buffer is flagged as full and execution stalled until the sample is read.

In the case of IIR filters, the Y buffer is used to store the set of M previous output samples, y[n-M] to y[n-1], used for calculating the next output sample y[n]. Each time a new sample is added to the set, the least recent sample y[n-M] drops out.

If the number of unread data in the buffer is less than the watermark threshold programmed in the EMPTY_WM bitfield of the FMAC_YBUFCFG register, the buffer is flagged as empty. As long as the empty flag is not set, interrupts or DMA requests are generated, if enabled, to request reads from the buffer. The watermark allows several data to be transferred under one interrupt, without danger of underflow. Nevertheless, if an underflow does occur, the UNFL error flag is set. In this case, the read pointer is not incremented and the read operation returns the content of the memory at the read pointer address.

The operation of the Y buffer in circular mode is illustrated in *Figure 104*. This example shows a 7-tap IIR filter with a watermark set to four.

Figure 104. Circular output buffer operation



21.3.5 Initialization functions

The following functions initialize the FMAC unit. They are triggered by writing the appropriate value in the FUNC bitfield of the FMAC_PARAM register, with the START bit set. The P and Q bitfields must also contain the appropriate parameter values for each function as detailed below. The R bitfield is not used. When the function completes, the START bit is automatically reset by hardware.

During initialization, it is recommended that the DMA requests and interrupts be disabled. The transfer of data into the FMAC memory can be done by software or by memory-to-memory DMA transfers, since no flow control is required.

Load X1 buffer

This function pre-loads the X1 buffer with N values, starting from the address in X1_BASE. Successive writes to the FMAC_WDATA register load the write data into the X1 buffer and increment the write address. The write pointer points to the address X1_BASE + N when the function completes.

The function can be used to pre-load the buffer with the elements of a vector, or to initialize the input storage elements of a filter.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X1 buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

Load X2 buffer

This function pre-loads the X2 buffer with N + M values, starting from the address in X2_BASE. Successive writes to the FMAC_WDATA register load the write data into the X2 buffer and increment the write address.

The function can be used to pre-load the buffer with the elements of a vector, or the coefficients of a filter. In the case of an IIR, the N feed-forward and M feed-back coefficients are concatenated and loaded together into the X2 buffer. The total number of coefficients is equal to N + M. For an FIR, there are no feedback coefficients, so M = 0.

Parameters

- The parameter P contains the number of values, N, to be loaded into the X2 buffer starting from address X2_BASE.
- The parameter Q contains the number of values, M, to be loaded into the X2 buffer starting from address X2_BASE + N.
- The parameter R is not used.

The function completes when N + M writes have been performed to the FMAC_WDATA register.

Load Y buffer

This function pre-loads the Y buffer with N values, starting from the address in Y_BASE. Successive writes to the FMAC_WDATA register load the write data into the Y buffer and increment the write address. The read pointer points to the address Y_BASE + N when the function completes.

The function can be used to pre-load the feedback storage elements of an IIR filter.

Parameters

- The parameter P contains the number of values to be loaded into the Y buffer.
- The parameters Q and R are not used.

The function completes when N writes have been performed to the FMAC_WDATA register.

21.3.6 Filter functions

The following filter functions are supported by the FMAC unit. These functions are triggered by writing the corresponding value in the FUNC bitfield of the FMAC_PARAM register with the START bit set. The P, Q and R bitfields must also contain the appropriate parameter values for each function as detailed below. The filter functions continue to run until the START bit is reset by software.

Convolution (FIR filter)

Y = **B***__X__

$$y_n = 2^R \cdot \sum_{k=0}^{N} b_k x_{n-k}$$

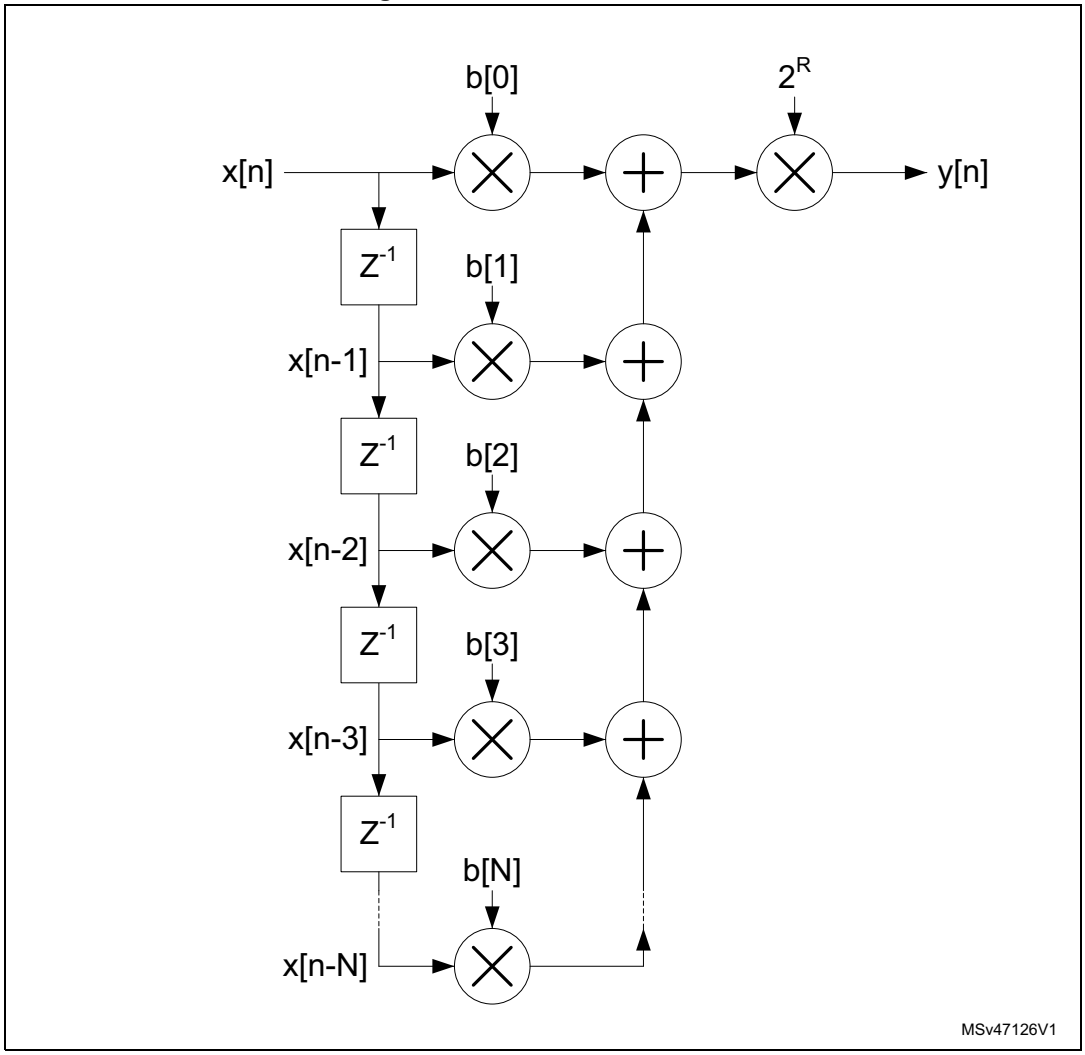
This function performs a convolution of a vector **B** of length N+1 and a vector **X** of indefinite length. The elements of **Y** for incrementing values of n are calculated as the dot product,

$y_n = \underline{\mathbf{B}}.\underline{\mathbf{X}}_{\mathbf{n}}$, where $\underline{\mathbf{X}}_{\mathbf{n}} = [x_{n-N},...,x_n]$ is composed of the N+1 elements of **X** at indexes n - N to n.

This function corresponds to a finite impulse response (FIR) filter, where vector **B** contains the filter coefficients and vector **X** the sampled data.

The structure of the filter (direct form) is shown in *Figure 105*.

Figure 105. FIR filter structure



Note that the cross correlation vector can be calculated by reversing the order of the coefficient vector **B**.

Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1 + d.
- X2 buffer contains the elements of vector **B**. It is a fixed buffer of length N + 1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length d.

Parameters:

- The parameter P contains the length, N+1, of the coefficient vector **B** in the range [2:127].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7]
- The parameter Q is not used.

The function completes when the START bit in the FMAC_PARAM register is reset by software.

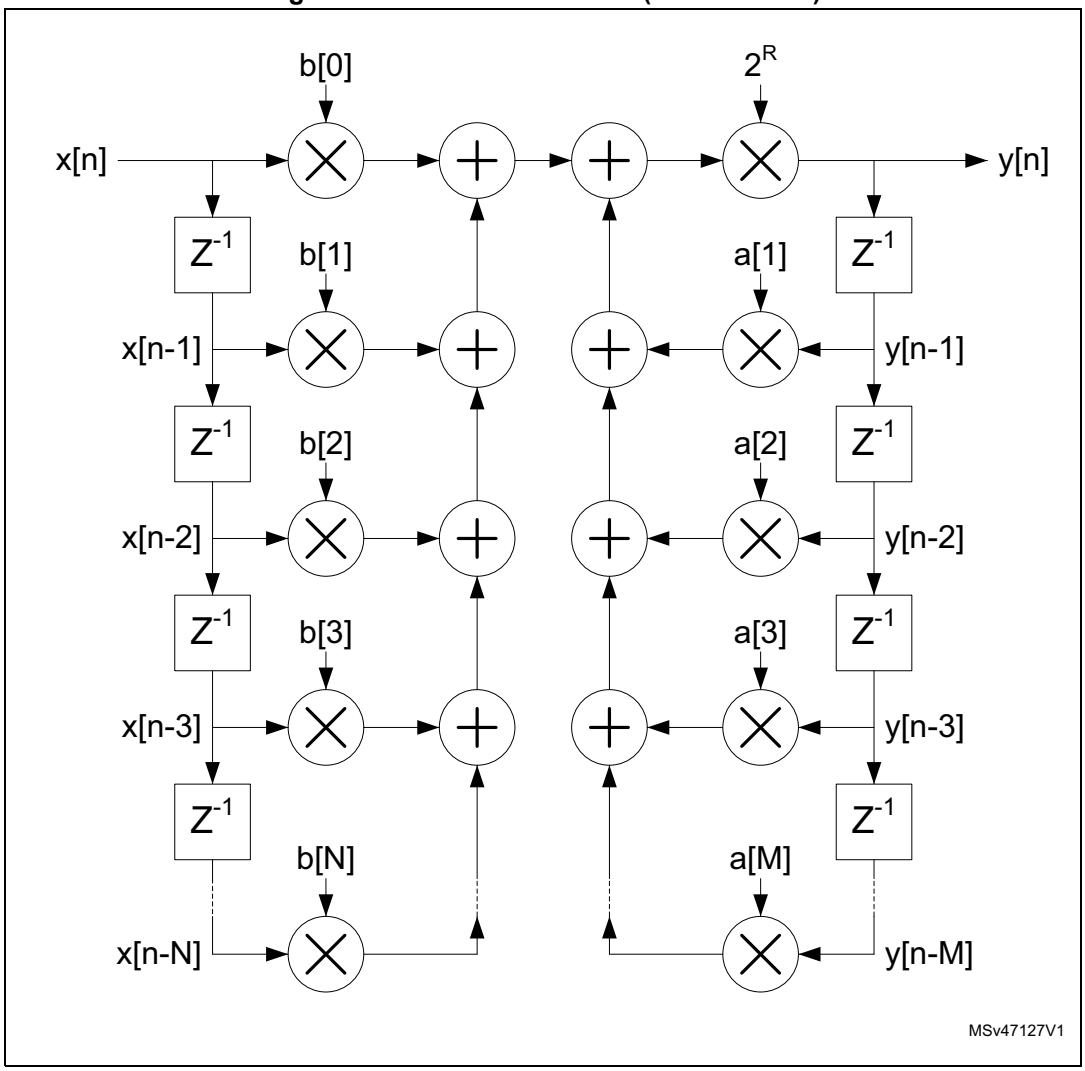
IIR filter

Y = B*X+A*Y'

$$y_n = 2^R \cdot \left(\sum_{k=0}^{N} b_k x_{n-k} + \sum_{k=1}^{M} a_k y_{n-k} \right)$$

This function implements an infinite impulse response (IIR) filter. The filter output vector **Y** is the convolution of a coefficient vector **B** of length N+1 and a vector **X** of indefinite length, plus the convolution of the delayed output vector **Y'** with a second coefficient vector **A**, of length M. The elements of **Y** for incrementing values of n are calculated as y_n = **B**.$\mathbf{X_n}$ + **A**.$\mathbf{Y_{n-1}}$, where $\mathbf{X_n}$ = $[x_{n-N},...,x_n]$ comprises the N+1 elements of **X** at indexes n - N to n, while $\mathbf{Y_{n-1}}$ = $[y_{n-M},...,y_{n-1}]$ comprises the M elements of **Y** at indexes n - M to n - 1. The structure of the filter (direct form 1) is shown in *Figure 106*.

Figure 106. IIR filter structure (direct form 1)



Input:

- X1 buffer contains the elements of vector **X**. It is a circular buffer of length N + 1+ d.
- X2 buffer contains the elements of coefficient vectors **B** and **A** concatenated (b_0, b_1, b_2..., b_N, a_1, a_2, ..., a_M). It is a fixed buffer of length M+N+1.

Output:

- Y buffer contains the output values, y_n. It is a circular buffer of length M + d.

Parameters

- The parameter P contains the length, N + 1, of the coefficient vector **B** in the range [2:64].
- The parameter Q contains the length, M, of the coefficient vector **A** in the range [1:63].
- The parameter R contains the gain to be applied to the accumulator output. The value output to the Y buffer is multiplied by 2^R, where R is in the range [0:7].

The function completes when the START bit in the FMAC_PARAM register is reset by software.

21.3.7 Fixed point representation

The FMAC operates in fixed point signed integer format. Input and output values are q1.15.

In q1.15 format, numbers are represented by one sign bit and 15 fractional bits (binary decimal places). The numeric range is therefore -1 (0x8000) to 1 - 2^{-15} (0x7FFF).

The accumulator has 26 bits, of which 22 are fractional and 4 are integer/sign (q4.22). This allows it to support partial accumulation sums in the range -8 (0x2000000) to +7.99999976 (0x1FFFFFF). A programmable gain from 0dB to 42dB in steps of 6dB can be applied at the output of the accumulator.

Note that the content of the accumulator is not saturated if the numeric range is exceeded. Partial sums whose value is greater than +7.99999976 or less than -8, wrap but this is harmless provided subsequent accumulations undo the wrapping. Nevertheless, the SAT flag in the FMAC_SR register is set if wrapping occurs, and generates an interrupt if the SATIEN bit is set in the FMAC_CR register. This helps in debugging the filter.

The data output by the accumulator can optionally be saturated, after application of the programmable gain, by setting the CLIPEN bit in the FMAC_CR register. If this bit is set, then any value which exceeds the numeric range of the q1.15 output, is set to 1 - 2^{-15} or -1, according to the sign. If clipping is not enabled, the unused accumulator bits after applying the gain is simply truncated.

21.3.8 Implementing FIR filters with the FMAC

The FMAC supports FIR filters of length N, where N is the number of taps or coefficients. The minimum local memory requirement for a FIR filter of length N is 2N + 1:

- N coefficients
- N input samples
- 1 output sample

Since the local memory size is 256, the maximum value for N is 127.

If maximum throughput is required, it may be necessary to allocate a small amount of extra space, d1 and d2, to the input and output sample buffers respectively, to ensure that the filter never stalls waiting for a new input sample, or waiting for the output sample to be read. In this case, the local memory requirement is 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N;
- Y_BUF_SIZE = d2 (or 1 if no extra space is required)

The buffer base addresses can be allocated anywhere, but the X2 buffer must not overlap with the others, or else the coefficients are overwritten. An example configuration is:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = 2N + d1

However, if the memory space is limited, the X1 and Y buffer areas can be overlapped, such that each output sample takes the place of the oldest input sample, which is no longer required:

- X2_BASE = 0;
- X1_BASE = N;
- Y_BASE = N

In this case, Y_BUF_SIZE = X1_BUF_SIZE = N + d1, so that the buffers remain in sync.

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. There is no point in pre-loading the Y buffer, since for the FIR filter there is no feedback path.

After configuring and initializing the buffers, the FMAC_CR register must be programmed according to the method used for writing and reading data to and from the FMAC memory.

Three methods are supported:

- Polling: No DMA request or Interrupt request is generated. Software must check that the X1_FULL flag is low before writing to WDATA, or that the Y_EMPTY flag is low before reading from RDATA.
- Interrupt: The interrupt request is asserted while the X1_FULL flag is low, for writes, or when the Y_EMPTY flag is low, for reads.
- DMA: DMA requests are asserted on the DMA write channel while the X1_FULL flag is low, and on the read channel while the Y_EMPTY flag is low.

Different methods can be used for read and for write. However it is not recommended to use both interrupts and DMA requests for the same operation[(a)]. The valid combinations are listed in *Table 172*.

Table 172. Valid combinations for read and write methods

WIEN	RIEN	DMAWEN	DMAREN	Write	Read
0	0	0	0	Polling	Polling
0	1	0	0	Polling	Interrupt
1	0	0	0	Interrupt	Polling
1	1	0	0	Interrupt	Interrupt
0	0	0	1	Polling	DMA
0	0	1	0	DMA	Polling
0	0	1	1	DMA	DMA
0	1	1	0	DMA	Interrupt
1	0	0	1	Interrupt	DMA

a. If both interrupts and DMA requests are enabled then only DMA must perform the transfer.

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 8 (FIR filter);
- P = N (number of coefficients);
- Q = “Don’t care”;
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples).

When 2^{EMPTY_WM} output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high.

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

21.3.9 Implementing IIR filters with the FMAC

The FMAC supports IIR filters of length N, where N is the number of feed-forward taps or coefficients. The number of feedback coefficients, M, can be any value from 1 to N-1. Only direct form 1 implementations can be realized, so filters designed for other forms need to be converted.

The minimum memory requirement for an IIR filter with N feed-forward coefficients and M feed-back coefficients is 2N + 2M:

- N + M coefficients
- N input samples
- M output samples

If M = N-1, then the maximum filter length that can be implemented is N = 64.

As for the FIR, for maximum throughput, a small amount of additional space, d1 and d2, is allowed in the input and output buffer size respectively, making the total memory requirement 2M + 2N + d1 + d2.

The buffers must be configured as follows:

- X1_BUF_SIZE = N + d1;
- X2_BUF_SIZE = N + M;
- Y_BUF_SIZE = M + d2;

The buffer base addresses can be allocated anywhere, but must not overlap. An example configuration is given below:

- X2_BASE = 0;
- X1_BASE = N + M;
- Y_BASE = 2N + M + d1;

Note: The FULL_WM bitfield of X1 buffer configuration register must be programmed with a value less than or equal to $log_2(d1)$, otherwise the buffer is flagged full before N input samples have been written, and no more samples are requested. Similarly, the EMPTY_WM bitfield of the Y buffer configuration register must be less than or equal to $log_2(d2)$.

The filter coefficients (N feed-forward followed by M feedback) **must** be pre-loaded into the X2 buffer, using the Load X2 Buffer function. The X1 buffer can optionally be pre-loaded with any number of samples up to a maximum of N. The Y buffer can optionally be pre-loaded with any number of values up to a maximum of M. This has the effect of initializing the feedback delay line.

After configuring the buffers, the FMAC_CR register must be programmed in the same way as for the FIR filter (see *Section 21.3.8: Implementing FIR filters with the FMAC*).

The filter is started by writing to the FMAC_PARAM register with the following bitfield values:

- FUNC = 9 (IIR filter);
- P = N (number of feed-forward coefficients);
- Q = M (number of feed-back coefficients);
- R = Gain;
- START = 1;

If less than N + d - 2^{FULL_WM} values have been pre-loaded in the X1 buffer, the X1FULL flag remains low. If the WIEN bit is set in the FMAC_CR register, then the interrupt request is asserted immediately to request the processor to write 2^{FULL_WM} additional samples into the buffer, via the FMAC_WDATA register. It remains asserted until the X1FULL flag goes high in the FMAC_SR register. The interrupt service routine must check the X1FULL flag after every 2^{FULL_WM} writes to the FMAC_WDATA register, and repeat the transfer until the flag goes high. Similarly, if the DMAWEN bit is set in the FMAC_CR register, DMA write channel requests are generated until the X1FULL flag goes high.

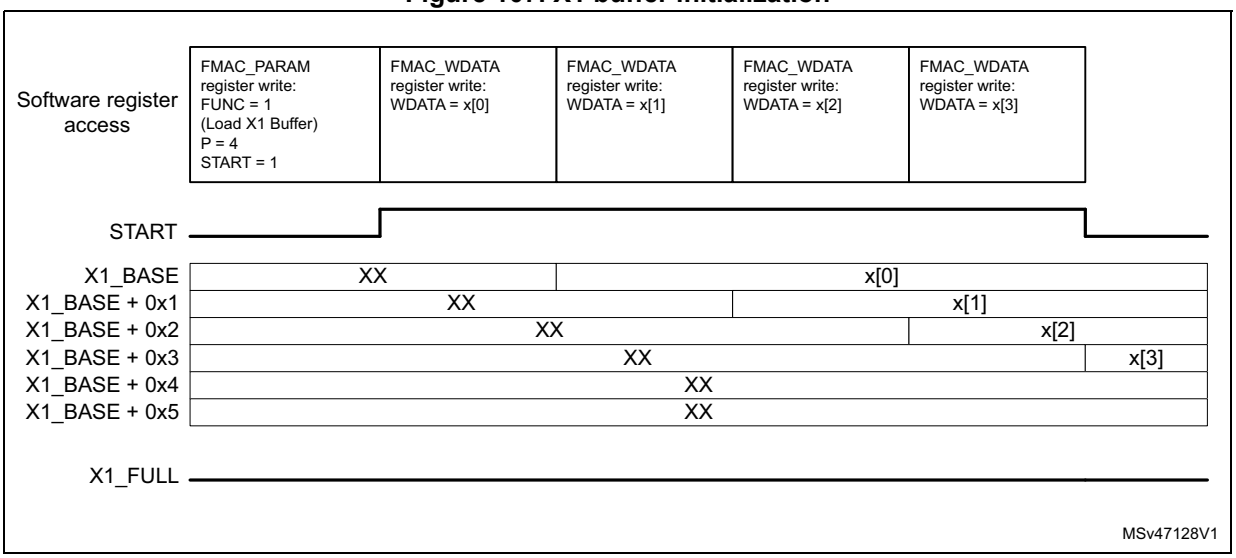
The filter calculates the first output sample when at least N samples have been written into the X1 buffer (including any pre-loaded samples). The first sample is calculated using the first N samples in the X1 buffer, and the first M samples in the Y buffer (whether or not they are preloaded. The first output sample is written into the Y buffer at Y_BASE + M.

When 2^{EMPTY_WM} new output samples have been written into the Y buffer, the YEMPTY flag in the FMAC_SR register goes low. If the RIEN bit is set in the FMAC_CR register, the interrupt request is asserted to request the processor to read 2^{EMPTY_WM} samples from the buffer, via the FMAC_RDATA register. It remains asserted until the YEMPTY flag goes high. The interrupt service routine must check the YEMPTY flag after every 2^{EMPTY_WM} reads from the FMAC_RDATA register, and repeat the transfer until the flag goes high. If the DMAREN bit is set in the FMAC_CR, DMA read channel requests are generated until the YEMPTY flag goes high

The filter continues to operate in this fashion until it is stopped by the software resetting the START bit.

21.3.10 Examples of filter initialization

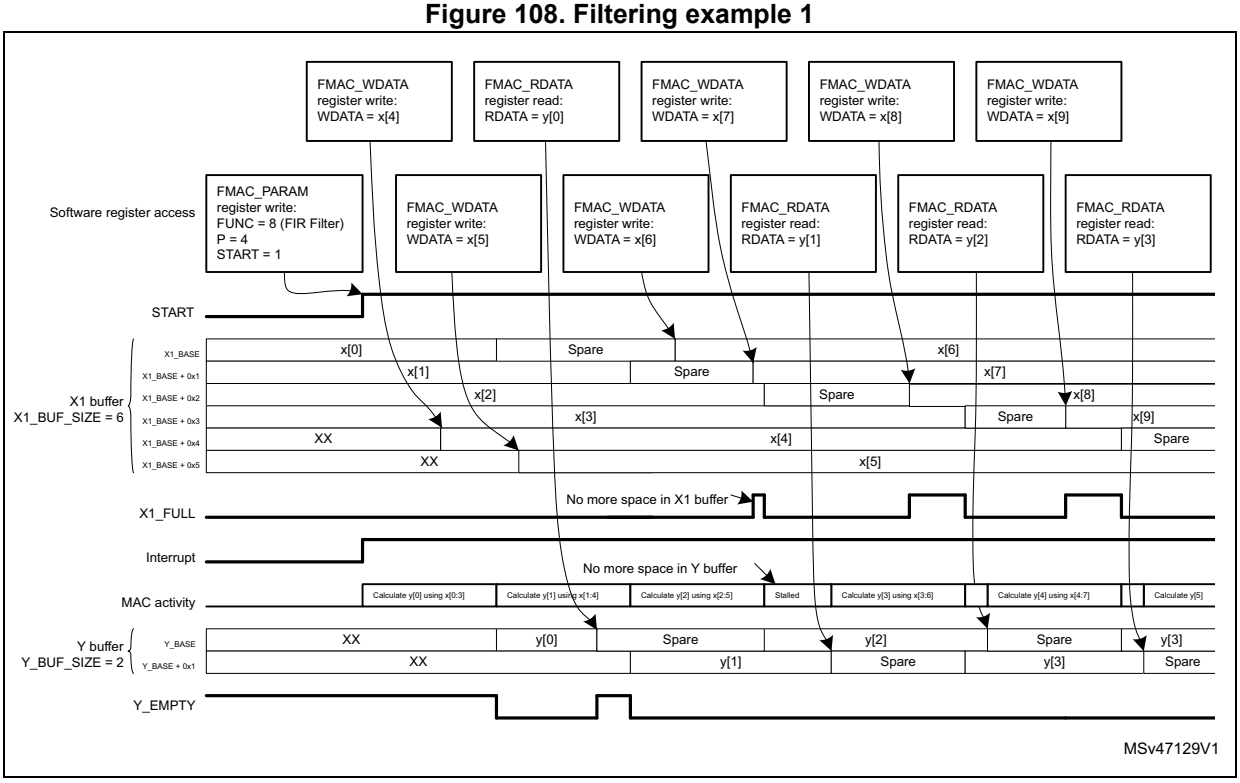
Figure 107. X1 buffer initialization



The example in *Figure 107* illustrates an X1 buffer pre-load with four samples (P = 4). The buffer size is six (X1_BUF_SIZE = 6). The initialization is launched by programming the FMAC_PARAM register with the START bit set. The four samples are then written to FMAC_WDATA, and transferred into local memory from X1_BASE onwards. The START bit resets after the fourth sample has been written. At this point, the X1 buffer contains the four samples, in order of writing, and the write pointer (next empty space) is at X1_BASE + 0x4.

21.3.11 Examples of filter operation

Figure 108. Filtering example 1



The example in *Figure 108* illustrates the beginning of a filter operation. The filter has four taps (P=4). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 107*. So the filter starts calculating the first output sample, y[0], immediately after the START bit is set. Since the X1FULL flag is not set (due to two uninitialized spaces in the X1 buffer), the interrupt is asserted straight away, to request new data. The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer.

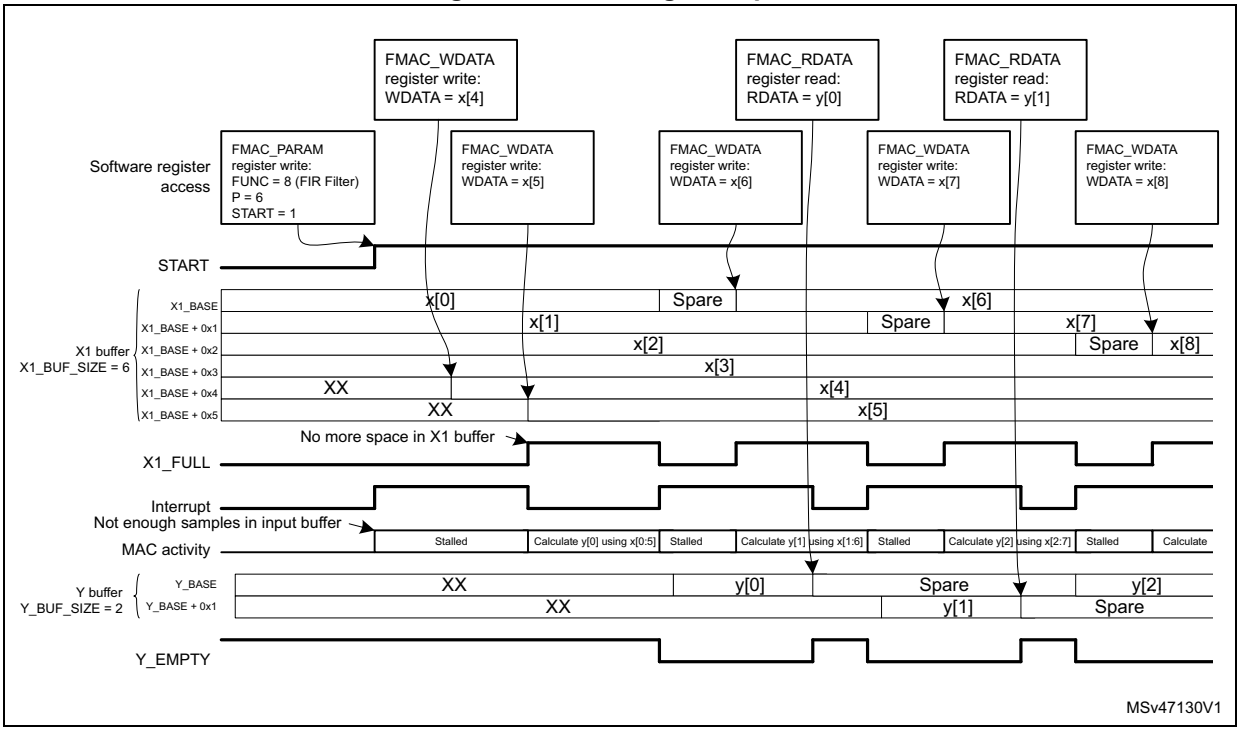
In the mean time, the FMAC finishes calculating the first output sample, y[0], and writes it into the Y buffer, causing the Y_EMPTY flag to go low. At the same time, the x[0] sample is discarded, as it is no longer required, freeing up its location in memory (at X1_BASE). The FMAC can immediately start work on the second output sample, y[1], since all the required input samples x[1:5] are present in the X1 buffer.

Since the Y_EMPTY flag is low, the interrupt remains active after the processor finishes writing x[5]. The processor reads y[0] from the FMAC_RDATA register, freeing up its location in the Y buffer. There are now no samples in the output buffer since y[1] is still being calculated, so the Y_EMPTY flag goes high. Nevertheless, the interrupt remains active, because there is still free space in the X1 buffer, which the processor next fills with x[6], and so on.

Note: In this example, the processor can fill the input buffer more quickly than the FMAC can process them, so the X1_full flag regularly goes active. However, it struggles to read the Y buffer fast enough, so the FMAC stalls regularly waiting for space to be freed up in the Y buffer. This means the filter is not executing at maximum throughput. The reason is that the

filter length is small and the processor relatively slow, in this example. So increasing the Y buffer size does not help.

Figure 109. Filtering example 2



The example in *Figure 109* illustrates the beginning of the same filter operation, but this time the filter has six taps (P=6). The X1 buffer size is six and the Y buffer size is two. The FULL_WM and EMPTY_WM bitfields are both set to 0. Prior to starting the filter, the X1 buffer has been pre-loaded with four samples, x[0:3] as in *Figure 107*. Because there are not enough samples in the input buffer, the X1FULL flag is not set, so the interrupt is asserted straight away, to request new data. The FMAC is stalled.

The processor writes two new samples, x[4] and x[5], to the FMAC_WDATA register, which are transferred to the empty locations in the X1 buffer. As soon as there are six unused samples in the X1 buffer, the X1_FULL flag goes active (since the buffer size is six), causing the interrupt to go inactive. The FMAC starts calculating the first output sample, y[0]. Since this requires all six input samples, there are no free spaces in the X1 buffer and so the X1_FULL flag remains active. Only when the FMAC finishes calculating y[0] and writes it into the Y buffer, can x[0] be discarded, freeing up a space in the X1 buffer, and deasserting X1_FULL. At the same time, the Y_EMPTY flag goes inactive. Both these flag states cause the interrupt to be asserted, requesting the processor to write a new input sample, first of all, and then read the output sample just calculated. The FMAC remains stalled until a new input sample is written.

In this example, the processor has to wait for the FMAC to finish calculating the current output sample, before it can write a new input sample, and therefore the X1 buffer regularly goes empty, stalling the FMAC. This can be avoided by allowing some extra space in the input buffer.

21.3.12 Filter design tips

The FMAC architecture imposes some constraints detailed below, on the design of digital filters.

1. Implementation of direct form 2, or transposed forms, is not efficient. Filters which have been designed for such forms must be converted to direct form 1.
2. Cascaded filters must either be combined into a single stage, or implemented as separate filters. In the latter case, multiple sets of filter coefficients can be pre-loaded into the memory, one set per stage, and only the X2_BASE address changed to select which set is used. The most efficient method of implementing a multi-stage filter is to pre-load a large X1 buffer with input samples, run the IIR filter function on it using the first stage coefficients, and store the output samples back in memory. Then change the X2_BASE pointer to point to the 2nd stage coefficients, and reload the input buffer with the output of the first stage (with a gain if required), before running the IIR function again. The procedure is repeated for all stages. Once the final stage samples have been transferred back into system memory, the input buffer can be loaded with the next set of input samples, and a new round of calculations started. Note that the N sample input buffer of each stage must be pre-loaded first of all with the N-1 last inputs from the previous round, plus one new sample, in order to keep continuity between each round. Similarly, the output buffer of each stage must be loaded with the last M samples from the previous round, for the same reason.
3. The use of direct form 1 for IIR designs can lead to large positive or negative partial sums in the accumulator, if for example a large step occurs on the input, or some of the filter coefficients' absolute values are >1. Since the accumulator is limited to 26 bits, the biggest value that it can handle without wrapping (changing sign) is 0x1FFFFFF positive or 0x2000000 negative. This corresponds to 3.99999988 and -4 respectively in q3.23 fixed point format. Wrapping does not represent a problem provided the wrapping is "undone" before the end of the accumulation. However this is not always the case when a filter is starting up and can lead to unexpected results. Consider pre-loading the output buffer with suitable values to avoid this.
4. The IIR filter has feed-forward (numerator) coefficients $[b_0, b_1, ..., b_{N-1}]$, and feed-back (denominator) coefficients $[1, a_1, ..., a_M]$. Many IIR filters require some of the denominator coefficients to have an absolute value greater than 1 to achieve a steep roll-off in the frequency response. Given that the coefficients are coded in fixed point q1.15 format, this is not possible. Nevertheless, by scaling the denominator coefficients by a factor 2^{-R}, such that $2^{-R}.[1, a_1, ..., a_M]$ are all less than 1, such filters can be implemented. However an inverse gain of 2^R must be applied at the output of the accumulator to compensate the scaling. This has an adverse effect on the signal-to-noise ratio.

21.4 FMAC registers

21.4.1 FMAC X1 buffer configuration register (FMAC_X1BUFCFG)

Address offset: 0x00

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:24 **FULL_WM[1:0]**: Watermark for buffer full flag

Defines the threshold for setting the X1 buffer full flag when operating in circular mode. The flag is set if the number of free spaces in the buffer is less than 2^{FULL_WM}.

0: Threshold = 1

1: Threshold = 2

2: Threshold = 4

3: Threshold = 8

Setting a threshold greater than 1 allows several data to be transferred into the buffer under one interrupt.

Threshold must be set to 1 if DMA write requests are enabled (DMAWEN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **X1_BUF_SIZE[7:0]**: Allocated size of X1 buffer in 16-bit words

The minimum buffer size is the number of feed-forward taps in the filter (+ the watermark threshold - 1).

Bits 7:0 **X1_BASE[7:0]**: Base address of X1 buffer

21.4.2 FMAC X2 buffer configuration register (FMAC_X2BUFCFG)

Address offset: 0x04

Reset value: 0x0000 0000

Access: word access

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **X2_BUF_SIZE[7:0]**: Size of X2 buffer in 16-bit words
This bitfield can not be modified when a function is ongoing (START = 1).

Bits 7:0 **X2_BASE[7:0]**: Base address of X2 buffer
The X2 buffer base address can be modified while START=1, for example to change coefficient values. The filter must be stalled when doing this, since changing the coefficients while a calculation is ongoing affects the result.

21.4.3 FMAC Y buffer configuration register (FMAC_YBUFCFG)

Address offset: 0x08

Reset value: 0x0000 0000

Access: word access

This register can only be modified if START = 0 in the FMAC_PARAM register.

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

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

Bits 25:24 **EMPTY_WM[1:0]**: Watermark for buffer empty flag
Defines the threshold for setting the Y buffer empty flag when operating in circular mode. The flag is set if the number of unread values in the buffer is less than 2^{EMPTY_WM}.
0: Threshold = 1
1: Threshold = 2
2: Threshold = 4
3: Threshold = 8
Setting a threshold greater than 1 allows several data to be transferred from the buffer under one interrupt.
Threshold must be set to 1 if DMA read requests are enabled (DMAREN = 1 in FMAC_CR register).

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:8 **Y_BUF_SIZE[7:0]**: Size of Y buffer in 16-bit words
For FIR filters, the minimum buffer size is 1 (+ the watermark threshold). For IIR filters the minimum buffer size is the number of feedback taps (+ the watermark threshold).

Bits 7:0 **Y_BASE[7:0]**: Base address of Y buffer

21.4.4 FMAC parameter register (FMAC_PARAM)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: word access

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

Bit 31 **START**: Enable execution
0: Stop execution
1: Start execution
Setting this bit triggers the execution of the function selected in the FUNC bitfield. Resetting it by software stops any ongoing function. For initialization functions, this bit is reset by hardware.

Bits 30:24 **FUNC[6:0]**: Function
0: Reserved
1: Load X1 buffer
2: Load X2 buffer
3: Load Y buffer
4 to 7: Reserved
8: Convolution (FIR filter)
9: IIR filter (direct form 1)
10 to 127: Reserved
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 23:16 **R[7:0]**: Input parameter R.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 15:8 **Q[7:0]**: Input parameter Q.
The value of this parameter is dependent on the function.
This bitfield can not be modified when a function is ongoing (START = 1)

Bits 7:0 **P[7:0]**: Input parameter P.
The value of this parameter is dependent on the function
This bitfield can not be modified when a function is ongoing (START = 1)

21.4.5 FMAC control register (FMAC_CR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLIP EN	Res.	Res.	Res.	Res.	Res.	DMA WEN	DMA REN	Res.	Res.	Res.	SAT IEN	UNFL IEN	OVFL IEN	WIEN	RIEN
rw						rw	rw				rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RESET**: Reset FMAC unit

This resets the write and read pointers, the internal control logic, the FMAC_SR register and the FMAC_PARAM register, including the START bit if active. Other register settings are not affected. This bit is reset by hardware.

0: Reset inactive

1: Reset active

Bit 15 **CLIPEN**: Enable clipping

0: Clipping disabled. Values at the output of the accumulator which exceed the q1.15 range, wrap.

1: Clipping enabled. Values at the output of the accumulator which exceed the q1.15 range are saturated to the maximum positive or negative value (+1 or -1) according to the sign.

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **DMAWEN**: Enable DMA write channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the X1 buffer is not full.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bit 8 **DMAREN**: Enable DMA read channel requests

0: Disable. No DMA requests are generated

1: Enable. DMA requests are generated while the Y buffer is not empty.

This bit can only be modified when START= 0 in the FMAC_PARAM register. A read returns the current state of the bit.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SATIEN**: Enable saturation error interrupts

0: Disabled. No interrupts are generated upon saturation detection.

1: Enabled. An interrupt request is generated if the SAT flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 3 **UNFLIEN**: Enable underflow error interrupts

0: Disabled. No interrupts are generated upon underflow detection.

1: Enabled. An interrupt request is generated if the UNFL flag is set

This bit is set and cleared by software. A read returns the current state of the bit.

Bit 2 **OVFLIEN**: Enable overflow error interrupts
0: Disabled. No interrupts are generated upon overflow detection.
1: Enabled. An interrupt request is generated if the OVFL flag is set
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 1 **WIEN**: Enable write interrupt
0: Disabled. No write interrupt requests are generated.
1: Enabled. An interrupt request is generated while the X1 buffer FULL flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 0 **RIEN**: Enable read interrupt
0: Disabled. No read interrupt requests are generated.
1: Enabled. An interrupt request is generated while the Y buffer EMPTY flag is not set.
This bit is set and cleared by software. A read returns the current state of the bit.

21.4.6 FMAC status register (FMAC_SR)

Address offset: 0x14

Reset value: 0x0000 0001

Access: word access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1 FULL	Y EMPTY
					r	r	r							r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **SAT**: Saturation error flag
Saturation occurs when the result of an accumulation exceeds the numeric range of the accumulator.
0: No saturation detected
1: Saturation detected. If the SATIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 9 **UNFL**: Underflow error flag
An underflow occurs when a read is made from FMAC_RDATA when no valid data is available in the Y buffer.
0: No underflow detected
1: Underflow detected. If the UNFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

Bit 8 **OVFL**: Overflow error flag
An overflow occurs when a write is made to FMAC_WDATA when no free space is available in the X1 buffer.
0: No overflow detected
1: Overflow detected. If the OVFLIEN bit is set, an interrupt is generated.
This flag is cleared by a reset of the unit.

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

Bit 1 **X1FULL**: X1 buffer full flag

The buffer is flagged as full if the number of available spaces is less than the FULL_WM threshold. The number of available spaces is the difference between the write pointer and the least recent sample currently in use.

0: X1 buffer not full. If the WIEN bit is set, the interrupt request is asserted until the flag is set. If DMAWEN is set, DMA write channel requests are generated until the flag is set.

1: X1 buffer full.

This flag is set and cleared by hardware, or by a reset.

Note: after the last available space in the X1 buffer is filled there is a delay of 3 clock cycles before the X1FULL flag goes high. To avoid any risk of overflow it is recommended to insert a software delay after writing to the X1 buffer before reading the FMAC_SR. Alternatively, a FULL_WM threshold of 2 can be used.

Bit 0 **YEMPTY**: Y buffer empty flag

The buffer is flagged as empty if the number of unread data is less than the EMPTY_WM threshold. The number of unread data is the difference between the read pointer and the current output destination address.

0: Y buffer not empty. If the RIEN bit is set, the interrupt request is asserted until the flag is set. If DMAREN is set, DMA read channel requests are generated until the flag is set.

1: Y buffer empty.

This flag is set and cleared by hardware, or by a reset.

Note: after the last sample is read from the Y buffer there is a delay of 3 clock cycles before the YEMPTY flag goes high. To avoid any risk of underflow it is recommended to insert a software delay after reading from the Y buffer before reading the FMAC_SR. Alternatively, an EMPTY_WM threshold of 2 can be used.

21.4.7 FMAC write data register (FMAC_WDATA)

Address offset: 0x18

Reset value: 0x0000 0000

Access: word access

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **WDATA[15:0]**: Write data

When a write access to this register occurs, the write data are transferred to the address offset indicated by the write pointer. The pointer address is automatically incremented after each write access.

21.4.8 FMAC read data register (FMAC_RDATA)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: word access

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RDATA[15:0]**: Read data

When a read access to this register occurs, the read data are the contents of the Y output buffer at the address offset indicated by the READ pointer. The pointer address is automatically incremented after each read access.

21.4.9 FMAC register map

Table 173.FMAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMAC_X1BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	FULL_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X1_BUF_SIZE[7:0]								X1_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	FMAC_X2BUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	X2_BUF_SIZE[7:0]								X2_BASE[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	FMAC_YBUFCFG	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY_WM[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Y_BUF_SIZE[7:0]								Y_BASE[7:0]							
	Reset value							0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	FMAC_PARAM	START	FUNC[6:0]							R[7:0]								Q[7:0]								P[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	FMAC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESET	CLIPEN	Res.	Res.	Res.	Res.	Res.	DMAWEN	DMAREN	Res.	Res.	Res.	SATIEN	UNFLIEN	OVFLIEN	WIEN	RIEN
	Reset value																0	0						0	0				0	0	0	0	0
0x14	FMAC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAT	UNFL	OVFL	Res.	Res.	Res.	Res.	Res.	Res.	X1FULL	YEMPTY
	Reset value																						0	0	0							0	1
0x18	FMAC_WDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	FMAC_RDATA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

22 Flexible memory controller (FMC)

22.1 Introduction

The flexible memory controller (FMC) includes three memory controllers:

- The NOR/PSRAM memory controller
- The NAND memory controller
- The Synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) controller (only available on STM32H562/563/573xx devices)

22.2 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, SDRAM memories, and NAND flash memory. Its main purposes are:

- to translate AHB transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique chip select. The FMC performs only one access at a time to an external device.

The main features of the FMC controller are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR flash memory/OneNAND flash memory
  - PSRAM (4 memory banks)
  - Ferroelectric RAM (FRAM)
  - NAND flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) memories
- Interface with parallel LCD modules, supporting Intel 8080 and Motorola 6800 modes.
- Burst mode support for faster access to synchronous devices such as NOR flash memory, PSRAM and SDRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16-bit wide data bus
- Independent chip select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM and SDRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth
- Cacheable Read FIFO with 6 x32-bit depth (6 x14-bit address tag) for SDRAM controller.

The Write FIFO is common to all memory controllers and consists of:

- a Write Data FIFO which stores the AHB data to be written to the memory (up to 32 bits) plus one bit for the AHB transfer (burst or not sequential mode)
- a Write Address FIFO which stores the AHB address (up to 28 bits) plus the AHB data size (up to 2 bits). When operating in burst mode, only the start address is stored except when crossing a page boundary (for PSRAM and SDRAM). In this case, the AHB burst is broken into two FIFO entries.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.

The FMC registers that define the external device type and associated characteristics are usually set at boot time and do not change until the next reset or power-up.

However, only a few bits can be changed on-the-fly:

- MBKEN, FMCEN, WEN bits in FMC_BCRx register
- ECCEN and PBKEN bits in the FMC_PCR register
- IFS, IRS and ILS bits in the FMC_SR register

Follow the below sequence to modify parameters while the FMC is enabled:

1. First disable the FMC controller to prevent further accesses to any memory controller while the register is modified.
2. Update all required configurations.
3. Enable the FMC controller again.

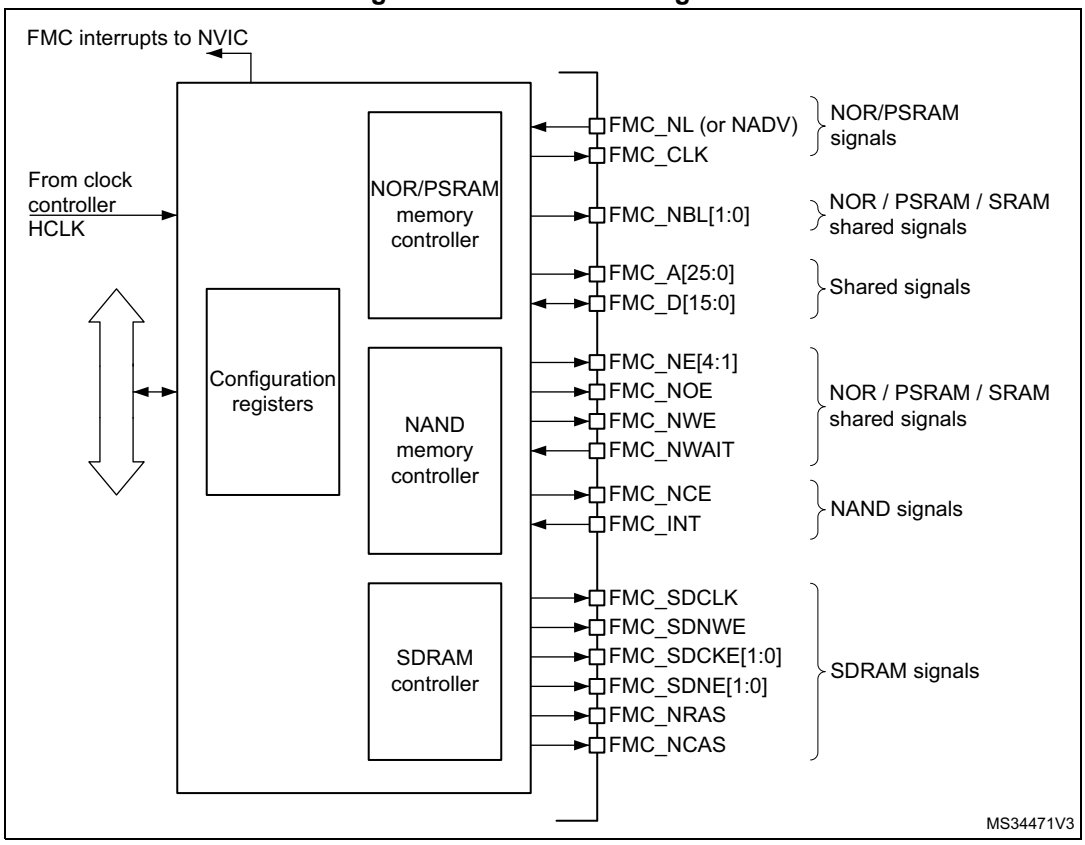
22.3 FMC block diagram

The FMC consists of the following main blocks:

- The AHB interface (including the FMC configuration registers)
- The NOR flash/PSRAM/SRAM controller
- The SDRAM controller

The block diagram is shown in the figure below.

Figure 110. FMC block diagram



22.4 AHB interface

The AHB slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AHB transactions are translated into the external device protocol. In particular, if the selected external memory is 16- or 8-bit wide, 32-bit wide transactions on the AHB are split into consecutive 16- or 8-bit accesses. The FMC chip select (FMC_NEx) does not toggle between the consecutive accesses except in case of Access mode D when the Extended mode is enabled.

The FMC generates an AHB error in the following conditions:

- When reading or writing to a FMC bank (Bank 1 to 4) which is not enabled.
- When reading or writing to the NOR flash bank while the FACCEN bit is reset in the FMC_BCRx register.
- When writing to a write protected SDRAM bank (WP bit set in the SDRAM_SDCRx register).
- When the SDRAM address range is violated (access to reserved address range)

The effect of an AHB error depends on the AHB master which has attempted the R/W access:

- If the access has been attempted by the Cortex-M33 CPU, a hard fault interrupt is generated.
- If the access has been performed by a DMA controller, a DMA transfer error is generated and the corresponding DMA channel is automatically disabled.

The AHB clock (HCLK) is the reference clock for the FMC.

22.4.1 Supported memories and transactions

General transaction rules

The requested AHB transaction data size can be 8-, 16- or 32-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AHB transaction size and memory data size are equal

  There is no issue in this case.
- AHB transaction size is greater than the memory size:

  In this case, the FMC splits the AHB transaction into smaller consecutive memory accesses to meet the external data width. The FMC chip select (FMC_NEx) does not toggle between the consecutive accesses. If the bus turnaround timings is configured to any other value than 0, the FMC chip select (FMC_NEx) toggles between the consecutive accesses. This feature is required when interfacing with FRAM memory.
- AHB transaction size is smaller than the memory size:

  The transfer may or not be consistent depending on the type of external device:
  - Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM, SDRAM)

    In this case, the FMC allows read/write transactions and accesses to the right data through its byte lanes NBL[1:0].

    Bytes to be written are addressed by NBL[1:0].

    All memory bytes are read (NBL[1:0] are driven low during read transaction) and the useless ones are discarded.
  - Accesses to devices that do not have the byte select feature (NOR and NAND flash memories)

    This situation occurs when a byte access is requested to a 16-bit wide flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the flash memory), Write transactions and Read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Wrap support for NOR flash/PSRAM and SDRAM

Wrap burst mode for synchronous memories is not supported. The memories must be configured in Linear burst mode of undefined length.

Configuration registers

The FMC can be configured through a set of registers. Refer to *Section 22.6.6*, for a detailed description of the NOR flash/PSRAM controller registers. Refer to *Section 22.7.7*,

for a detailed description of the NAND flash registers and to *Section 22.8.5* for a detailed description of the SDRAM controller registers.

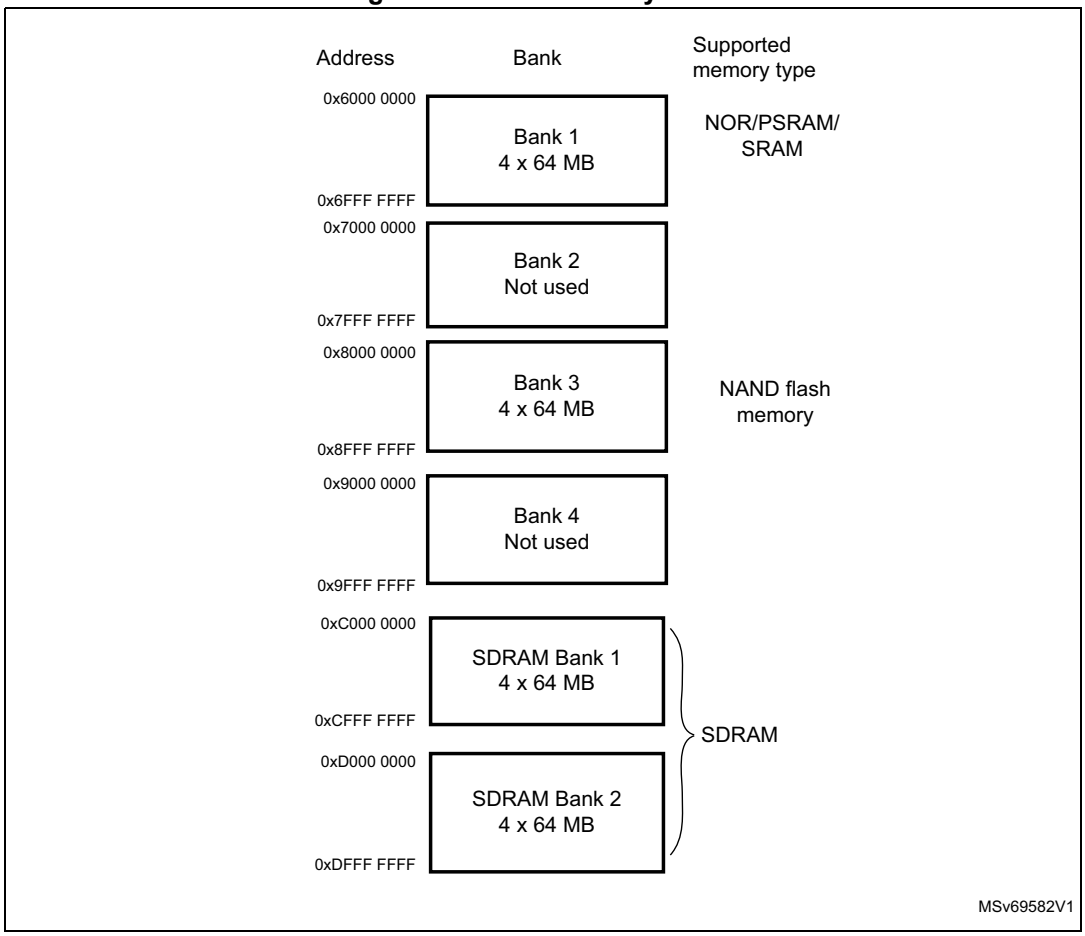
22.5 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see *Figure 111*):

- Bank 1 used to address up to 4 NOR flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated chip selects, as follows:
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 3 used to address NAND flash memory devices.The MPU memory attribute for this space must be reconfigured by software to Device.
- Bank 4 and 5 used to address SDRAM devices (1 device per bank).

For each bank the type of memory to be used can be configured by the user application through the Configuration register.

Figure 111. FMC memory banks



22.5.1 NOR/PSRAM address mapping

HADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 174*.

Table 174. NOR/PSRAM bank selection

HADDR[27:26](1)	Selected bank
00	Bank 1 - NOR/PSRAM 1
01	Bank 1 - NOR/PSRAM 2
10	Bank 1 - NOR/PSRAM 3
11	Bank 1 - NOR/PSRAM 4

1. HADDR are internal AHB address lines that are translated to external memory.

The HADDR[25:0] bits contain the external memory address. Since HADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 175. NOR/PSRAM External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	HADDR[25:0]	64 Mbytes x 8 = 512 Mbits
16-bit	HADDR[25:1] >> 1	64 Mbytes/2 x 16 = 512 Mbits

1. In case of a 16-bit external memory width, the FMC internally uses HADDR[25:1] to generate the address for external memory FMC_A[24:0].
Whatever the external memory width, FMC_A[0] must be connected to external memory address A[0].

22.5.2 NAND flash memory address mapping

The NAND bank is divided into memory areas as indicated in *Table 176*.

Table 176. NAND memory mapping and timing registers

Start address	End address	FMC bank	Memory space	Timing register
0x8800 0000	0x8BFF FFFF	Bank 3 - NAND flash	Attribute	FMC_PATT (0x8C)
0x8000 0000	0x83FF FFFF		Common	FMC_PMEM (0x88)

For NAND flash memory, the common and attribute memory spaces are subdivided into three sections (see in *Table 177* below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common / attribute memory space)
- Address section (next 128 Kbytes in the common / attribute memory space)

Table 177. NAND bank selection

Section name	HADDR[17:16]	Address range
Address section	1X	0x020000-0x03FFFF
Command section	01	0x010000-0x01FFFF
Data section	00	0x000000-0x0FFFF

The application software uses the 3 sections to access the NAND flash memory:

- **To sending a command to NAND flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data,** the software reads or writes the data from/to any memory location in the data section.

Since the NAND flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

22.5.3 SDRAM address mapping

The HADDR[28] bit (internal AHB address line 28) is used to select one of the two memory banks as indicated in *Table 178*.

Table 178. SDRAM bank selection

HADDR[28]	Selected bank	Control register	Timing register
0	SDRAM Bank1	FMC_SDCR1	FMC_SDTR1
1	SDRAM Bank2	FMC_SDCR2	FMC_SDTR2

The following table shows SDRAM mapping for a 13-bit row, a 11-bit column and a 4 internal bank configuration.

Table 179. SDRAM address mapping

Memory width(1)	Internal bank	Row address	Column address(2)	Maximum memory capacity (Mbytes)
8-bit	HADDR[25:24]	HADDR[23:11]	HADDR[10:0]	64 Mbytes: 4 x 8K x 2K
16-bit	HADDR[26:25]	HADDR[24:12]	HADDR[11:1]	128 Mbytes: 4 x 8K x 2K x 2

1. When interfacing with a 16-bit memory, the FMC internally uses the HADDR[11:1] internal AHB address lines to generate the external address. Whatever the memory width, FMC_A[0] has to be connected to the external memory address A[0].
2. The AutoPrecharge is not supported. FMC_A[10] must be connected to the external memory address A[10] but it is always driven low.

The HADDR[27:0] bits are translated to external SDRAM address depending on the SDRAM controller configuration:

- Data size:8 or 16 bits
- Row size:11, 12 or 13 bits
- Column size: 8, 9, 10 or 11 bits
- Number of internal banks: two or four internal banks

The following tables show the SDRAM address mapping versus the SDRAM controller configuration.

Table 180. SDRAM address mapping with 8-bit data bus width(1)(2)

Row size configuration	HADDR(AHB Internal Address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.							Bank [1:0]		Row[10:0]											Column[7:0]							
	Res.							Bank [1:0]		Row[10:0]											Column[8:0]							
	Res.							Bank [1:0]		Row[10:0]											Column[9:0]							
	Res.							Bank [1:0]		Row[10:0]											Column[10:0]							
12-bit row size configuration	Res.							Bank [1:0]		Row[11:0]											Column[7:0]							
	Res.							Bank [1:0]		Row[11:0]											Column[8:0]							
	Res.							Bank [1:0]		Row[11:0]											Column[9:0]							
	Res.							Bank [1:0]		Row[11:0]											Column[10:0]							
13-bit row size configuration	Res.							Bank [1:0]		Row[12:0]											Column[7:0]							
	Res.							Bank [1:0]		Row[12:0]											Column[8:0]							
	Res.							Bank [1:0]		Row[12:0]											Column[9:0]							
	Res.							Bank [1:0]		Row[12:0]											Column[10:0]							

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved (Res.) address range generates an AHB error.

Table 181. SDRAM address mapping with 16-bit data bus width(1)(2)

Row size Configuration	HADDR(AHB address Lines)																									
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]											Column[7:0]								BM0(3)
	Res.						Bank [1:0]		Row[10:0]											Column[8:0]								BM0
	Res.						Bank [1:0]		Row[10:0]											Column[9:0]								BM0
	Res.				Bank [1:0]		Row[10:0]													Column[10:0]								BM0
12-bit row size configuration	Res.						Bank [1:0]		Row[11:0]											Column[7:0]								BM0
	Res.						Bank [1:0]		Row[11:0]											Column[8:0]								BM0
	Res.				Bank [1:0]		Row[11:0]													Column[9:0]								BM0
	Res.						Bank [1:0]		Row[11:0]											Column[10:0]								BM0
13-bit row size configuration	Res.						Bank [1:0]		Row[12:0]											Column[7:0]								BM0
	Res.				Bank [1:0]		Row[12:0]													Column[8:0]								BM0
	Res.						Bank [1:0]		Row[12:0]											Column[9:0]								BM0
	Res.						Bank [1:0]		Row[12:0]											Column[10:0]								BM0

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved space (Res.) generates an AHB error.
3. BM0: is the byte mask for 16-bit access.

22.6 NOR flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed
- NOR flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique chip select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC Clock (FMC_CLK) is a submultiple of the HCLK clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in Synchronous mode (see *Section 22.6.6: NOR/PSRAM controller registers*). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AHB data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency is changed depending on AHB data transaction (refer to *Section 22.6.5: Synchronous transactions* for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see *Section 22.6.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 182*) and support for wait management (for PSRAM and NOR flash accessed in Burst mode).

Table 182. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	AHB clock cycle (HCLK)	1	15
NBL setup	Duration of the byte lanes setup phase	Asynchronous	AHB clock cycle (HCLK)	0	3
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	1	256
Data hold	Duration of the data hold phase	Asynchronous	AHB clock cycle (HCLK)	0	3
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read / write	AHB clock cycle (HCLK)	0	15

Table 182. Programmable NOR/PSRAM access parameters (continued)

Parameter	Function	Access mode	Unit	Min.	Max.
Clock divide ratio	Number of AHB clock cycles (HCLK) to build one memory clock cycle (CLK)	Synchronous	AHB clock cycle (HCLK)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (CLK)	2	17

22.6.1 External memory interface signals

Table 183, *Table 184* and *Table 185* list the signals that are typically used to interface with NOR flash memory, SRAM and PSRAM.

Note: The prefix "N" identifies the signals that are active low.

NOR flash memory, non-multiplexed I/Os

Table 183. Non-multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR flash memory, 16-bit multiplexed I/Os

Table 184. 16-bit multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip select, x = 1..4
NOE	O	Output enable
NWE	O	Write enable

Table 184. 16-bit multiplexed I/O NOR flash memory (continued)

FMC signal name	I/O	Function
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/FRAM/SRAM, non-multiplexed I/Os

Table 185. Non-multiplexed I/Os PSRAM/SRAM

FMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[15:0]	I/O	Data bidirectional bus
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (CellularRAM™ i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 186. 16-Bit multiplexed I/O PSRAM

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip select, x = 1..4 (called NCE by PSRAM (CellularRAM™ i.e. CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[1:0]	O	Byte lane output. Byte 0 and Byte 1 control (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

22.6.2 Supported memories and transactions

Table 187 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 187. NOR flash/PSRAM: example of supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NOR flash (muxed I/Os and nonmuxed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32	16	Y	-
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32	16	Y	-
	Synchronous	W	8	16	Y	Use of byte lanes NBL[1:0]
	Synchronous	W	16/32	16	Y	-
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	-
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[1:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses Use of byte lanes NBL[1:0]

22.6.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK)
- In Synchronous mode (read or write), all output signals change on the rising edge of HCLK. Whatever the CLKDIV value, all outputs change as follows:
  - NOEL/NWEL/ NEL/NADVL/ NADVH /NBLL/ Address valid outputs change on the falling edge of FMC_CLK clock.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

22.6.4 NOR flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR flash, PSRAM, SRAM, FRAM)

- Signals are synchronized by the internal clock HCLK. This clock is not issued to the memory
- The FMC always samples the data before de-asserting the NOE signal. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the Extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the Extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in mode 1 or mode 2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/FRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 112. Mode 1 read access waveforms

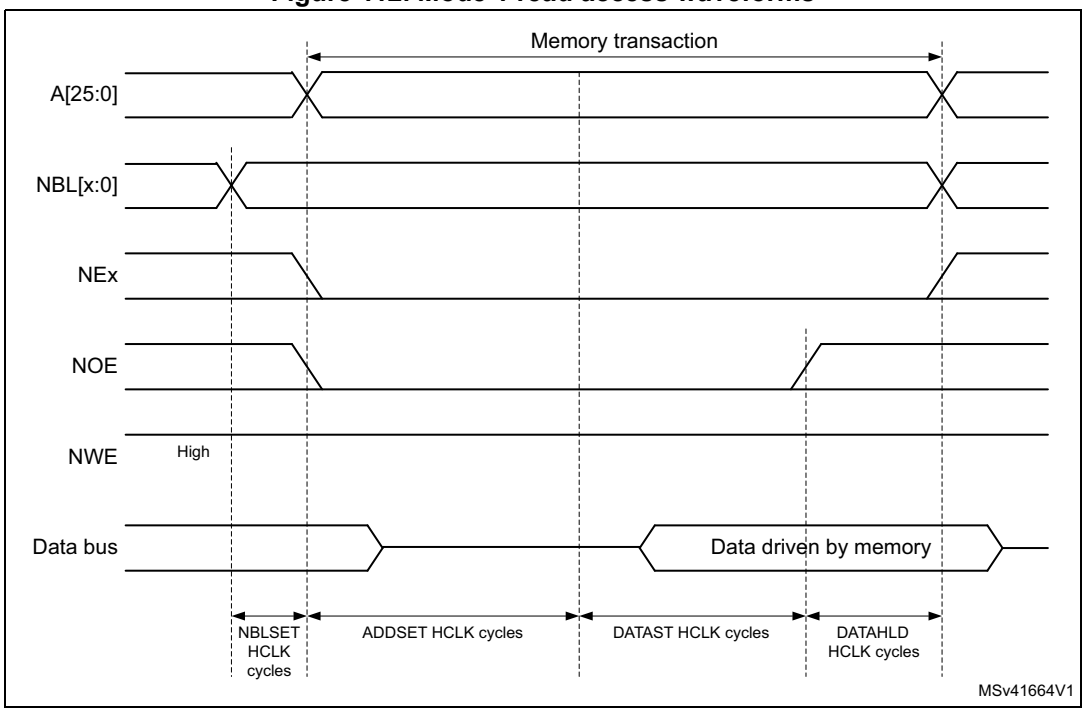
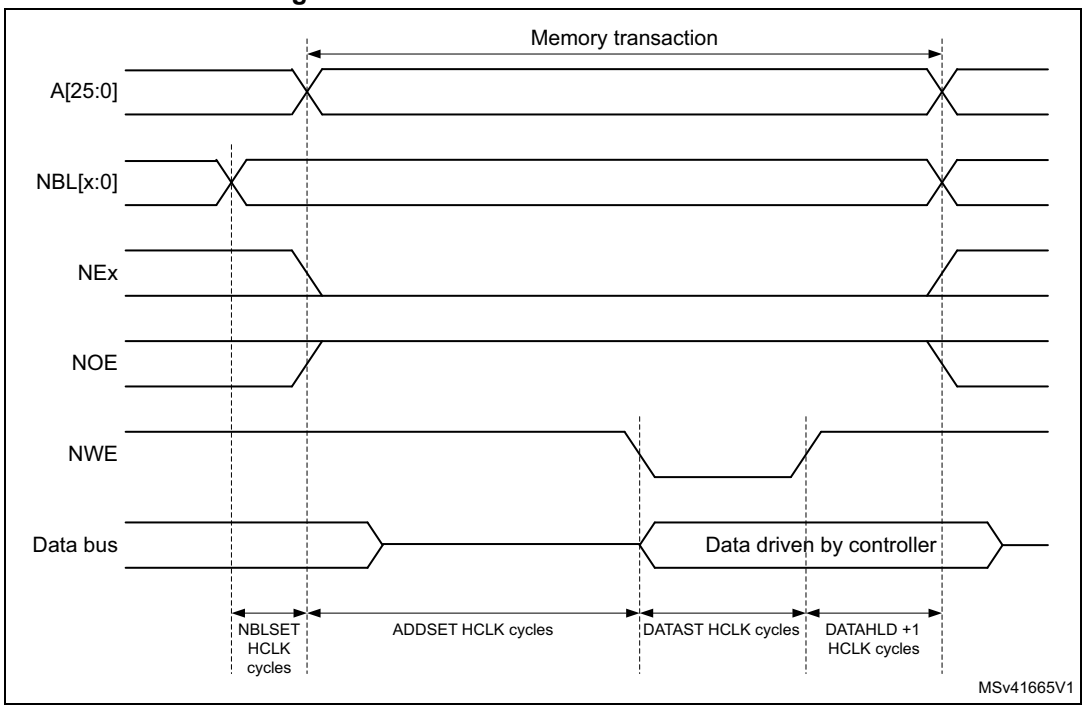


Figure 113. Mode 1 write access waveforms



The DATAHLD time at the end of the read and write transactions guarantee the address and data hold time after the NOE/NWE rising edge. The DATAST value must be greater than zero (DATAST > 0).

Table 188. FMC_BCRx bitfields (mode 1)

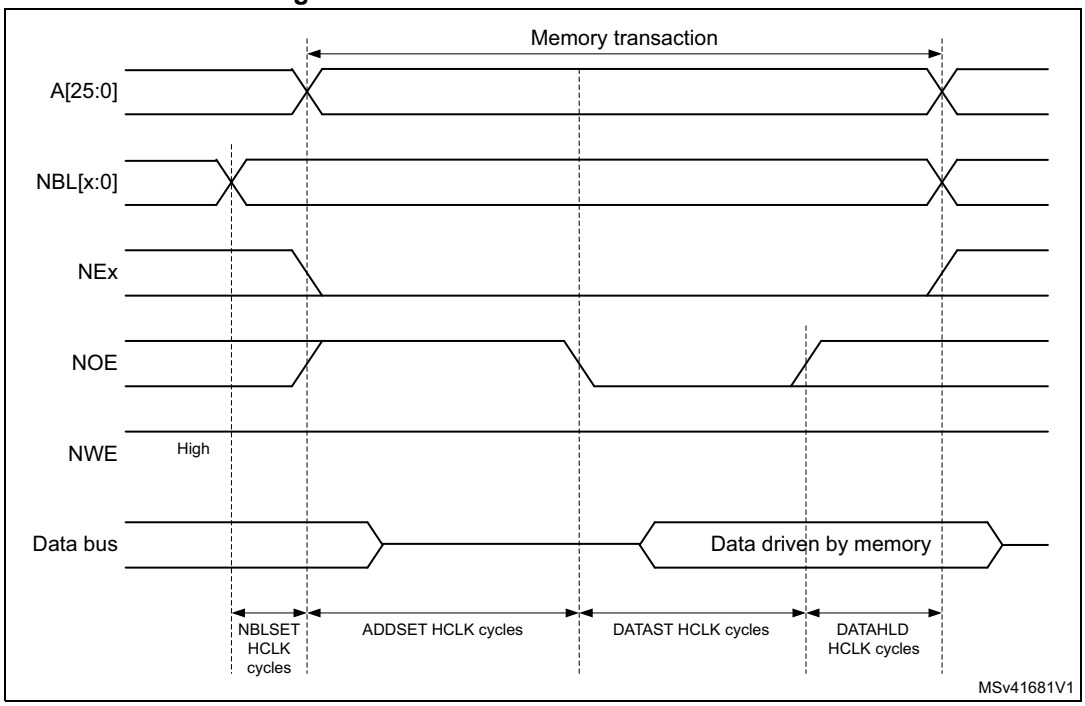
Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 189. FMC_BTRx bitfields (mode 1)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses, DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	Don’t care
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles).
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 0.

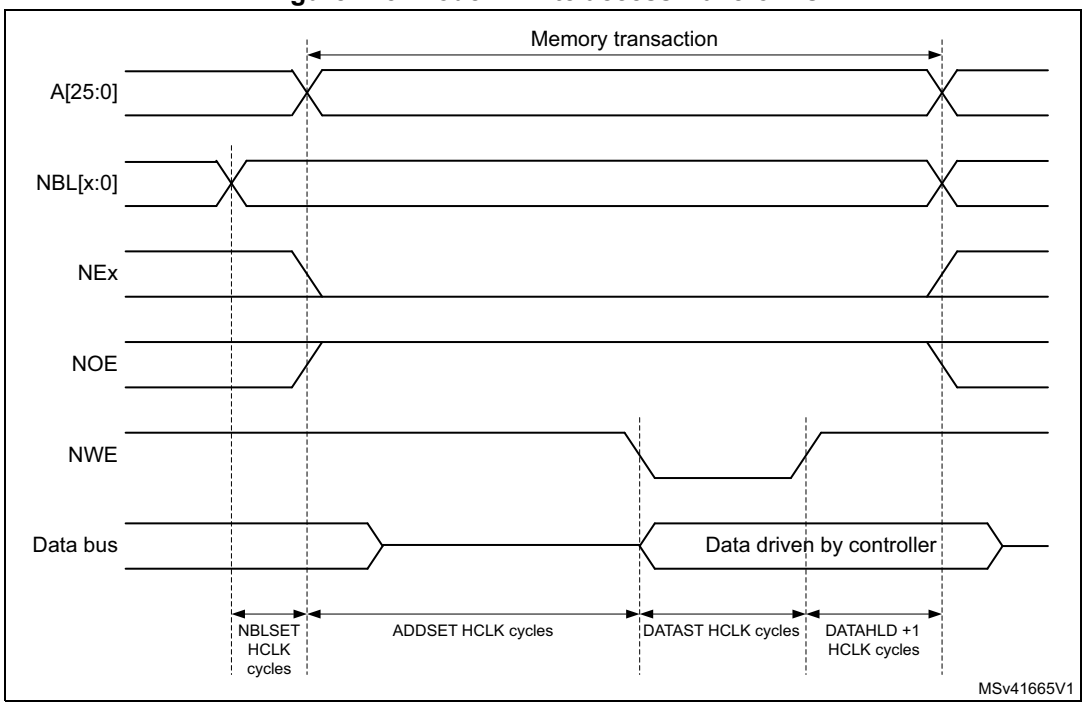
Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 114. Mode A read access waveforms



1. NBL[1:0] are driven low during the read access

Figure 115. Mode A write access waveforms



The differences compared with Mode 1 are the toggling of NOE and the independent read and write timings.

Table 190. FMC_BCRx bitfields (mode A)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	Don’t care
5:4	MWID	As needed
3:2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 191. FMC_BTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x0
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 192. FMC_BWTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x0
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR flash

Figure 116. Mode 2 and mode B read access waveforms

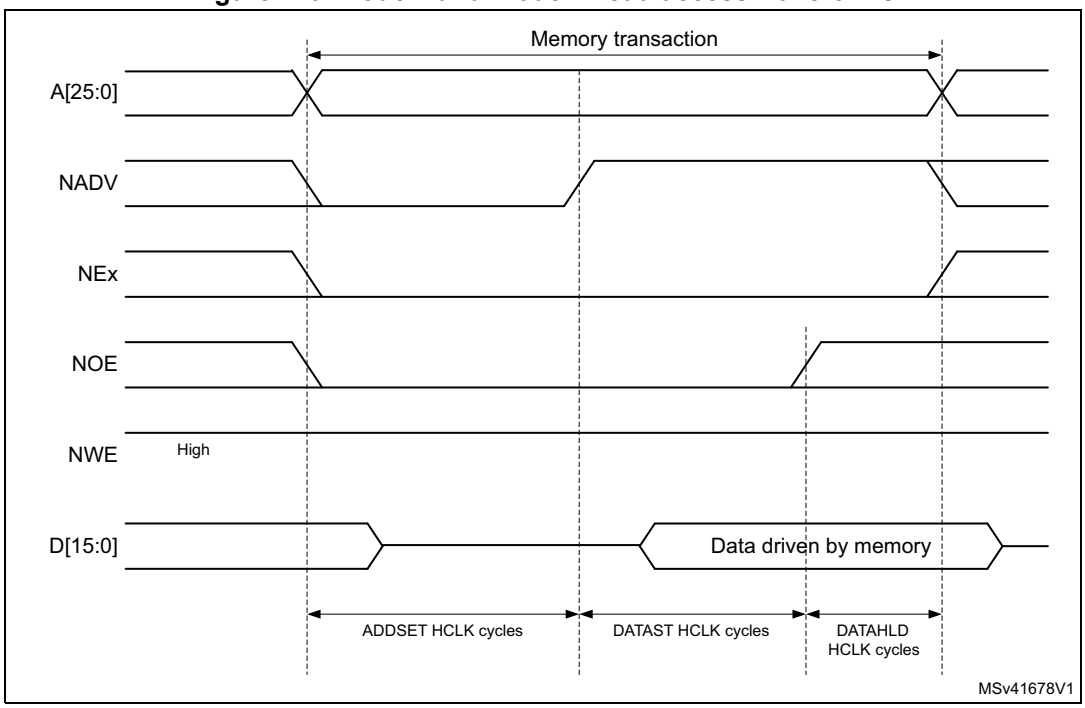


Figure 117. Mode 2 write access waveforms

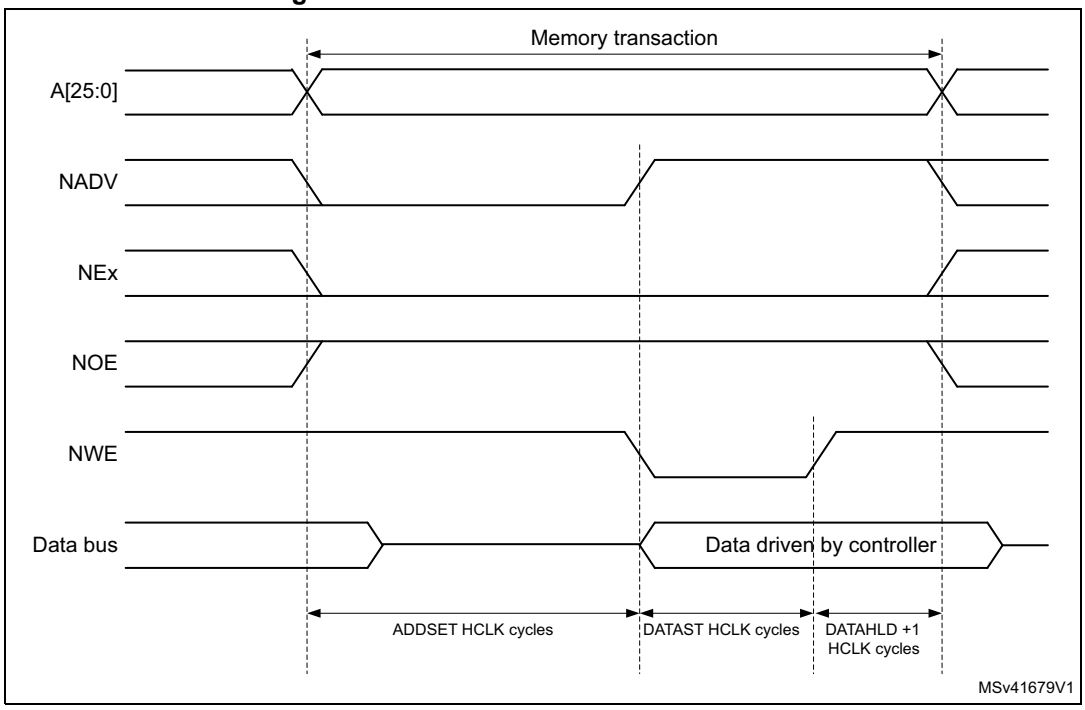
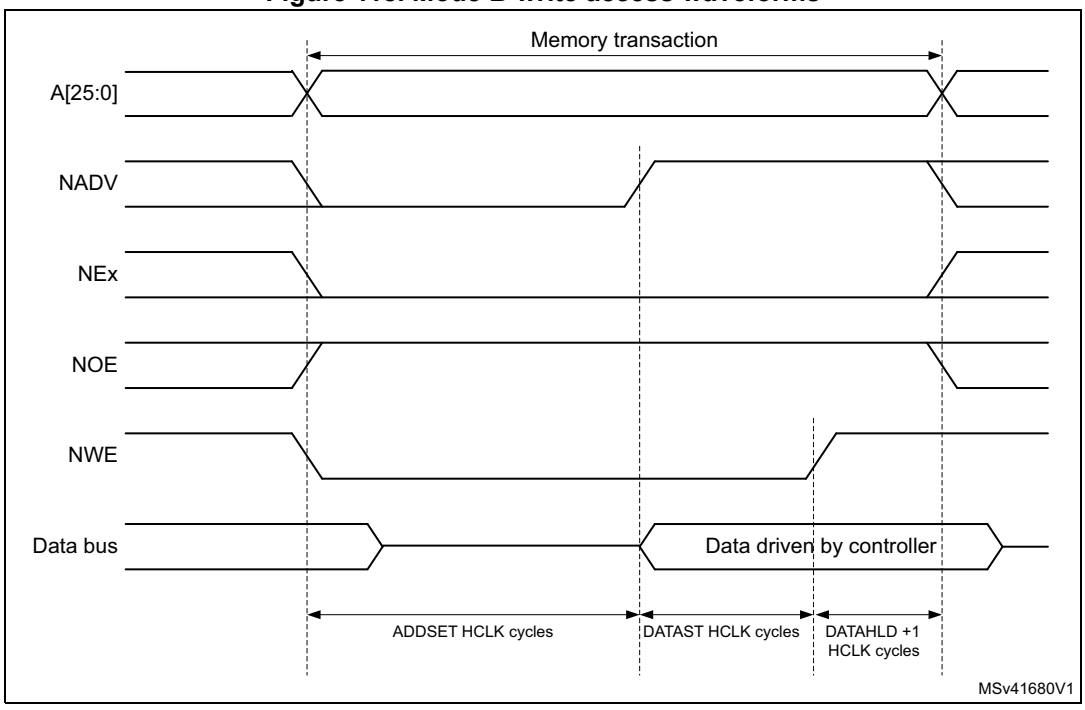


Figure 118. Mode B write access waveforms



The differences with mode 1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 193. FMC_BCRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	Don't care
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don't care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 194. FMC_BTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses and DATAHLD+1 HCLK cycles for write accesses when Extended mode is disabled).
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 195. FMC_BWTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x1 if Extended mode is set
27:24	DATLAT	Don't care
23:20	CLKDIV	Don't care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the access second phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don't care
3:0	ADDSET	Duration of the access first phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the Extended mode is set (mode B), otherwise its content is don't care.

Mode C - NOR flash - OE toggling

Figure 119. Mode C read access waveforms

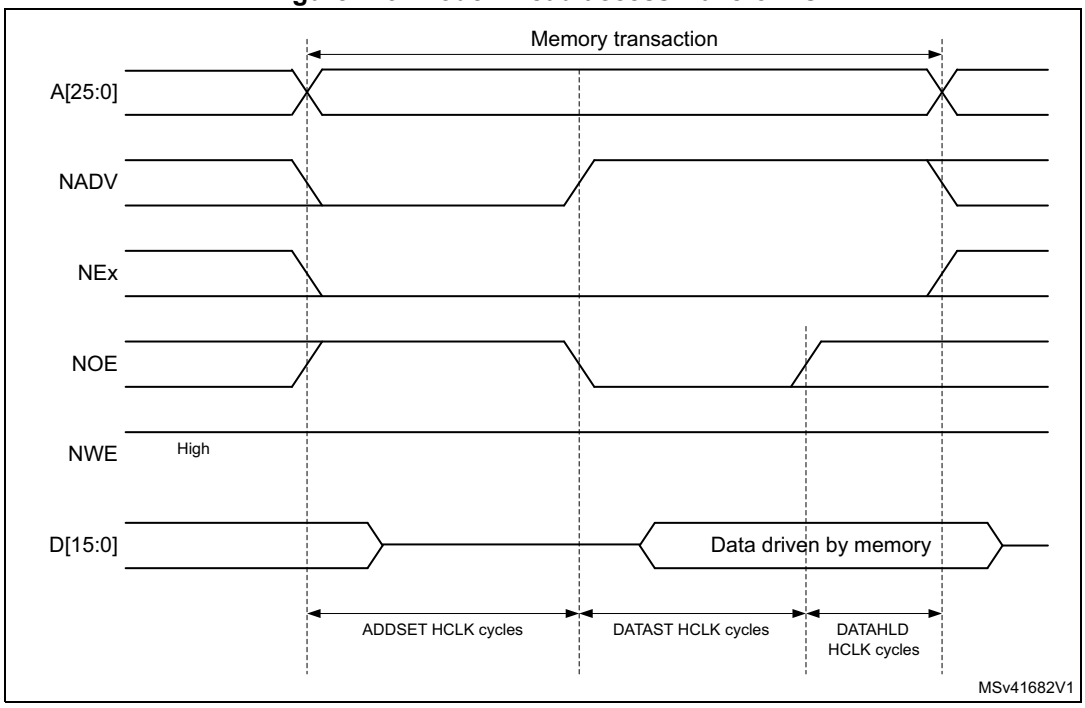
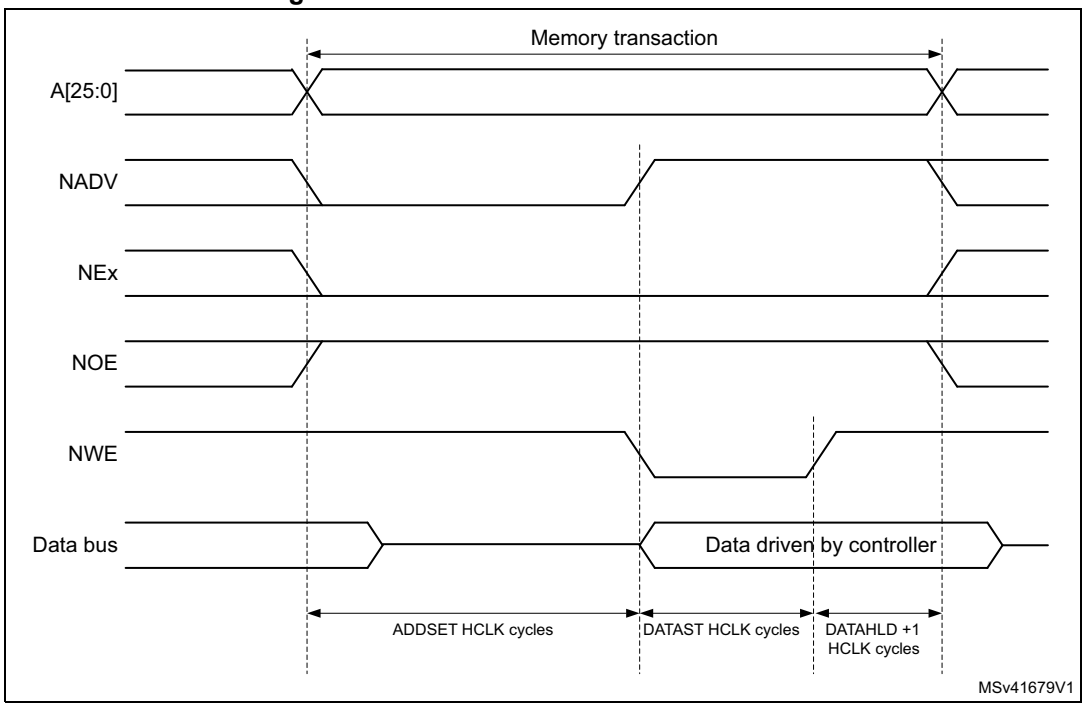


Figure 120. Mode C write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 196. FMC_BCRx bitfields (mode C)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	Don’t care
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	FACCEN	0x1

Table 196. FMC_BCRx bitfields (mode C) (continued)

Bit number	Bit name	Value to set
5:4	MWID	As needed
3:2	MTYP	0x02 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 197. FMC_BTRx bitfields (mode C)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x2
27:24	DATLAT	0x0
23:20	CLKDIV	0x0
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 0.

Table 198. FMC_BWTRx bitfields (mode C)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x2
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for write accesses.
7:4	ADDHLD	Don’t care
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 121. Mode D read access waveforms

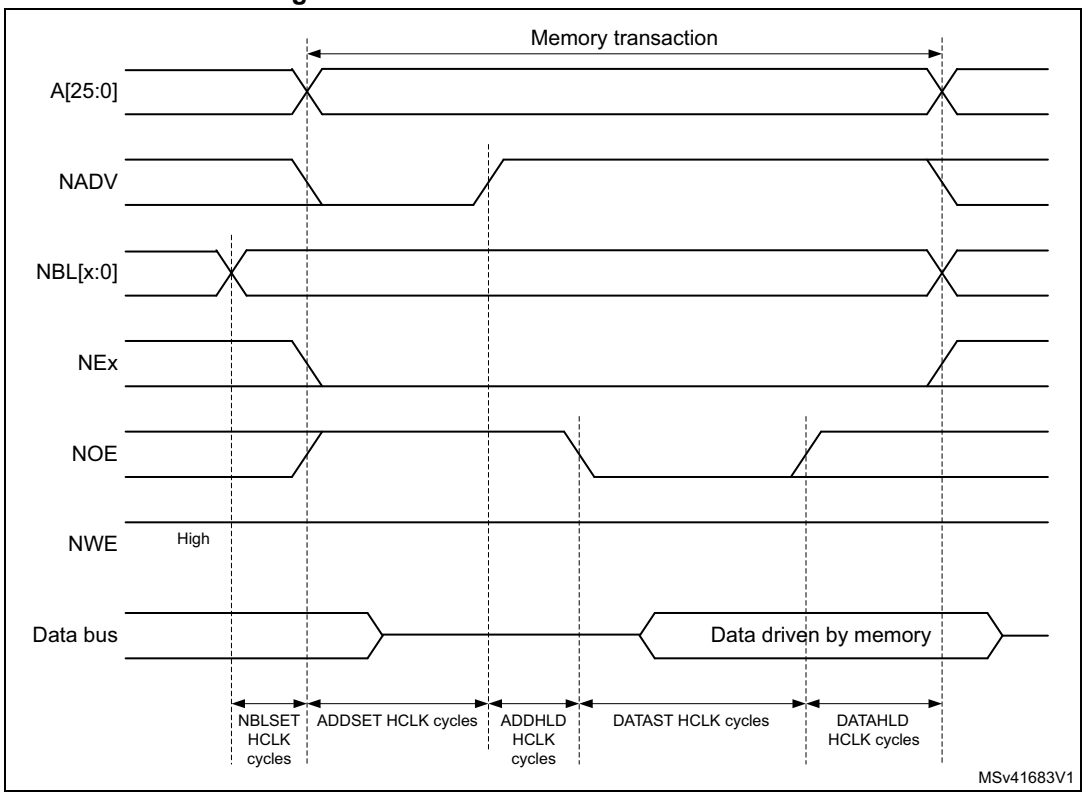
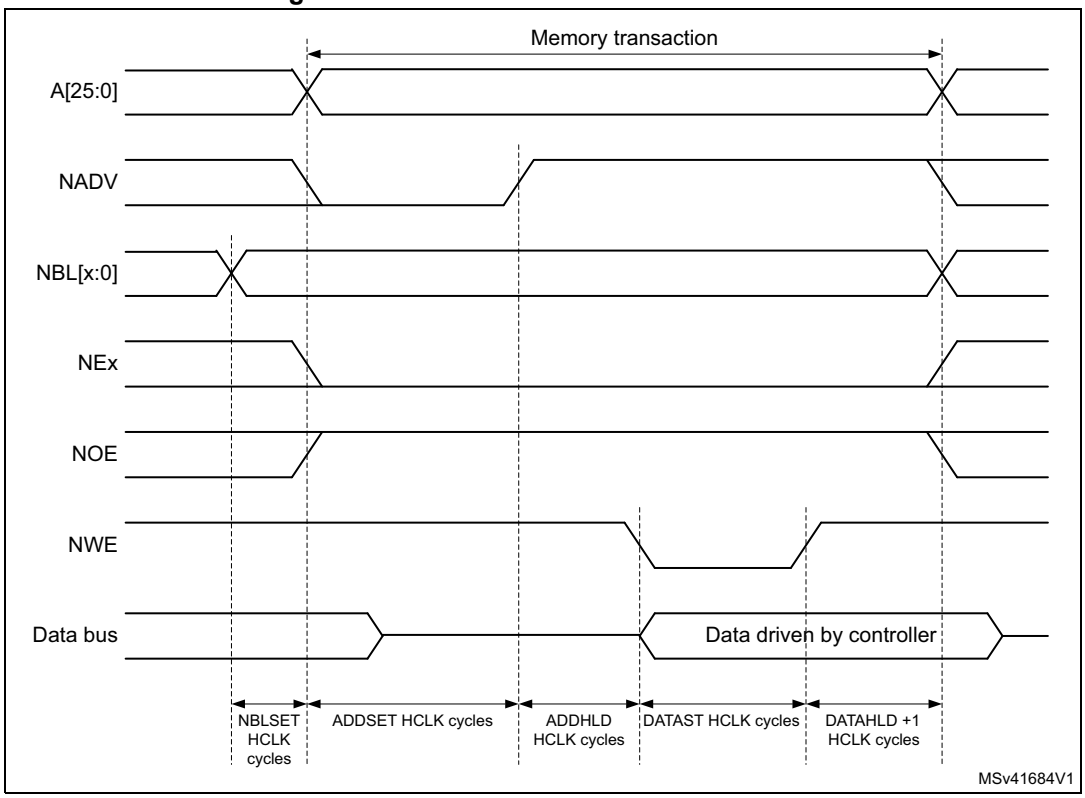


Figure 122. Mode D write access waveforms



The differences with mode 1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 199. FMC_BCRx bitfields (mode D)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0

Table 199. FMC_BCRx bitfields (mode D) (continued)

Bit number	Bit name	Value to set
7	Reserved	0x1
6	FACCEN	Set according to memory support
5:4	MWID	As needed
3:2	MTYP	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 200. FMC_BTRx bitfields (mode D)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses).
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles) for read accesses.
7:4	ADDHLD	Duration of the middle phase of the read access (ADDHLD HCLK cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for read accesses. Minimum value for ADDSET is 1.

Table 201. FMC_BWTRx bitfields (mode D)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x3
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles).
7:4	ADDHLD	Duration of the middle phase of the write access (ADDHLD HCLK cycles)
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR flash memory

Figure 123. Muxed read access waveforms

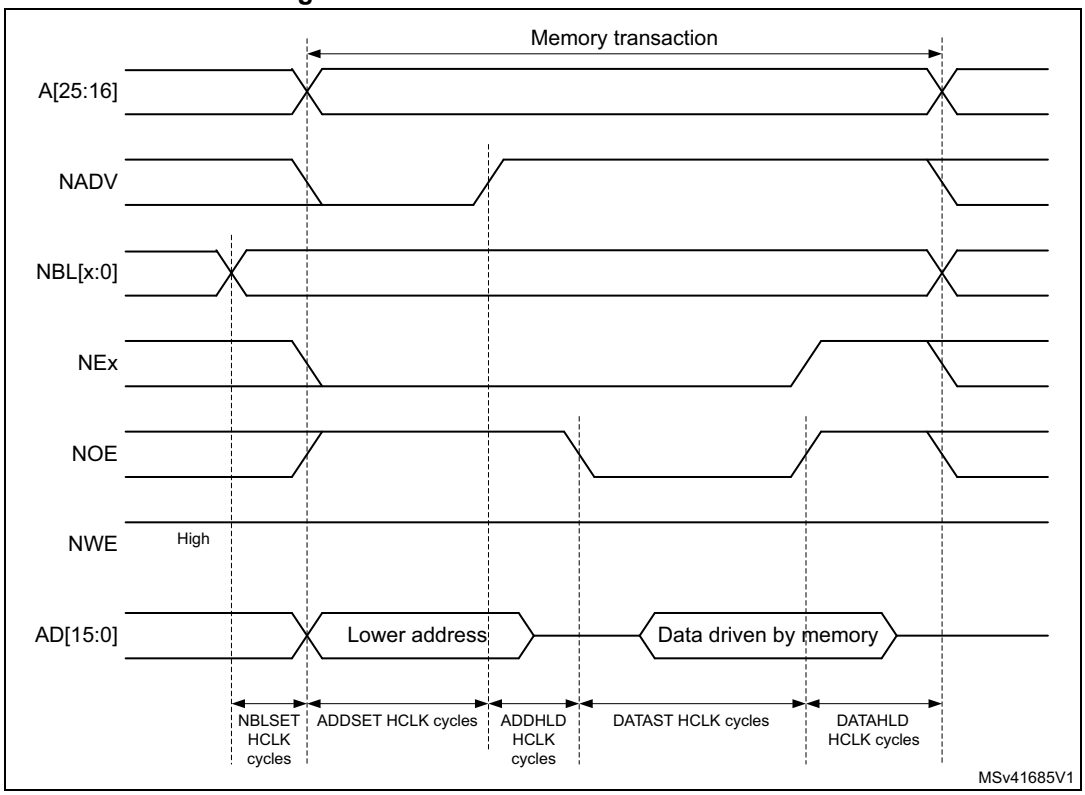
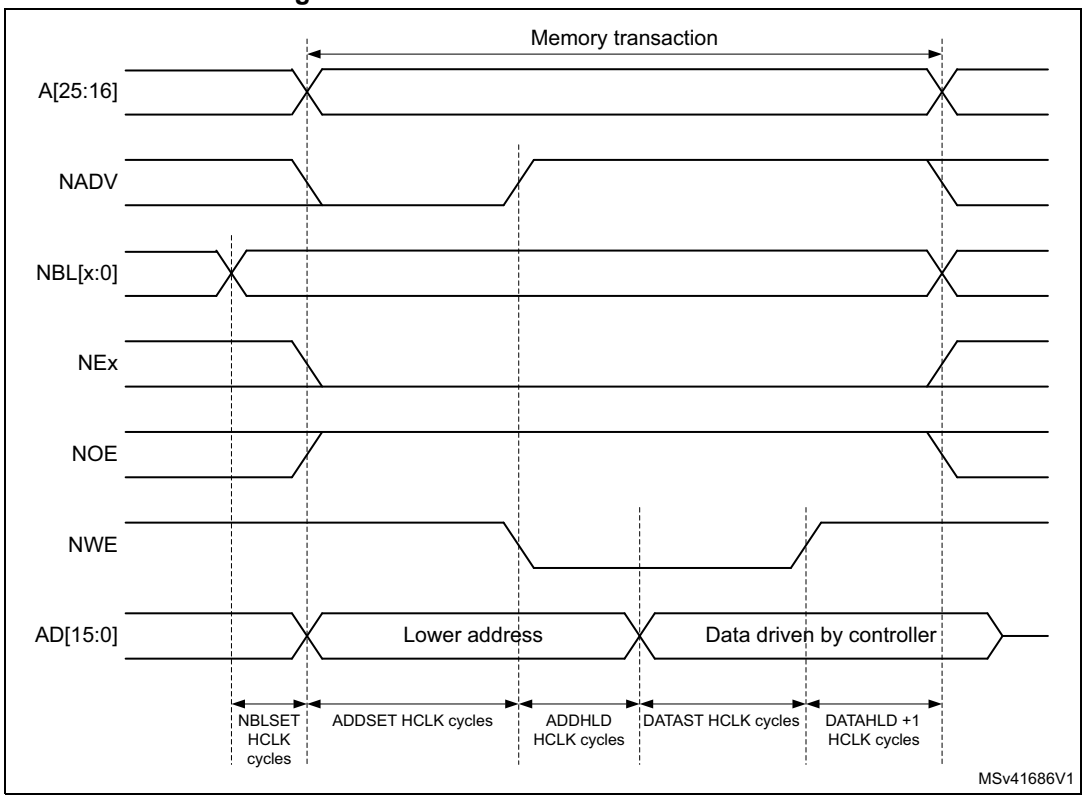


Figure 124. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 202. FMC_BCRx bitfields (Muxed mode)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
20	CCLKEN	As needed
19	CBURSTRW	0x0 (no effect in Asynchronous mode)
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	Set to 1 if the memory supports this feature. Otherwise keep at 0.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in Asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1

Table 202. FMC_BCRx bitfields (Muxed mode) (continued)

Bit number	Bit name	Value to set
6	FACCEN	0x1
5:4	MWID	As needed
3:2	MTYP	0x2 (NOR flash memory) or 0x1(PSRAM)
1	MUXEN	0x1
0	MBKEN	0x1

Table 203. FMC_BTRx bitfields (Muxed mode)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the data hold phase (DATAHLD HCLK cycles for read accesses, DATAHLD+1 HCLK cycles for write accesses).
29:28	ACCMOD	0x0
27:24	DATLAT	Don’t care
23:20	CLKDIV	Don’t care
19:16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15:8	DATAST	Duration of the second access phase (DATAST HCLK cycles).
7:4	ADDHLD	Duration of the middle phase of the access (ADDHLD HCLK cycles).
3:0	ADDSET	Duration of the first access phase (ADDSET HCLK cycles). Minimum value for ADDSET is 1.

WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 HCLK cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$DATAST \geq (4 \times HCLK) + max_wait_assertion_time$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):

   if

$$max_wait_assertion_time > address_phase + hold_phase$$

   then:

$$DATAST \geq (4 \times HCLK) + (max_wait_assertion_time - address_phase - hold_phase)$$

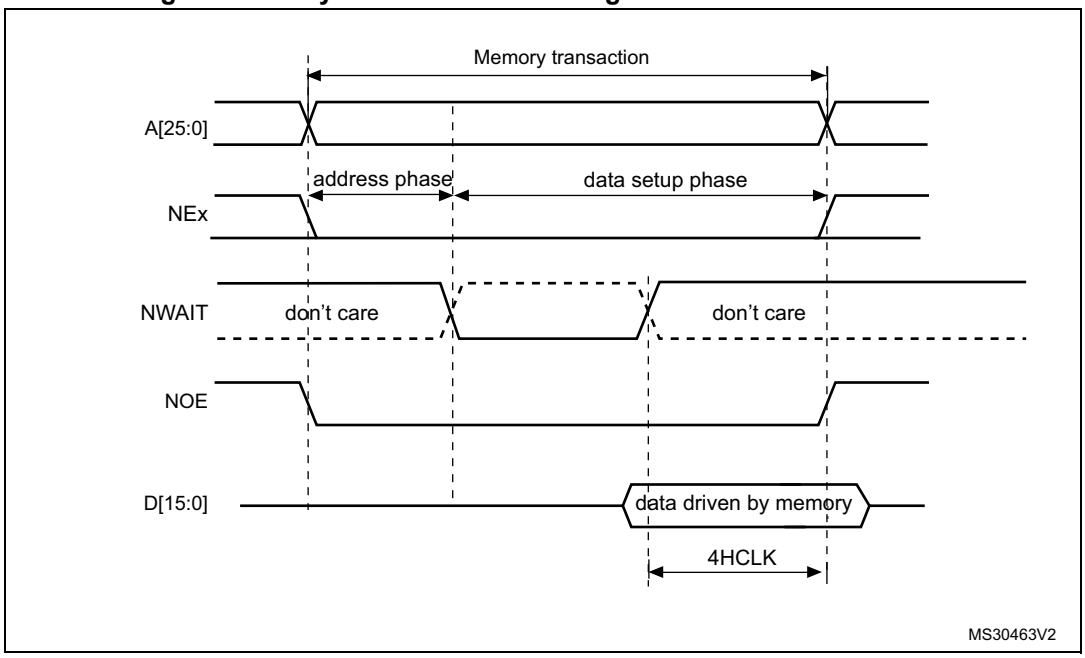
   otherwise

$$DATAST \geq 4 \times HCLK$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

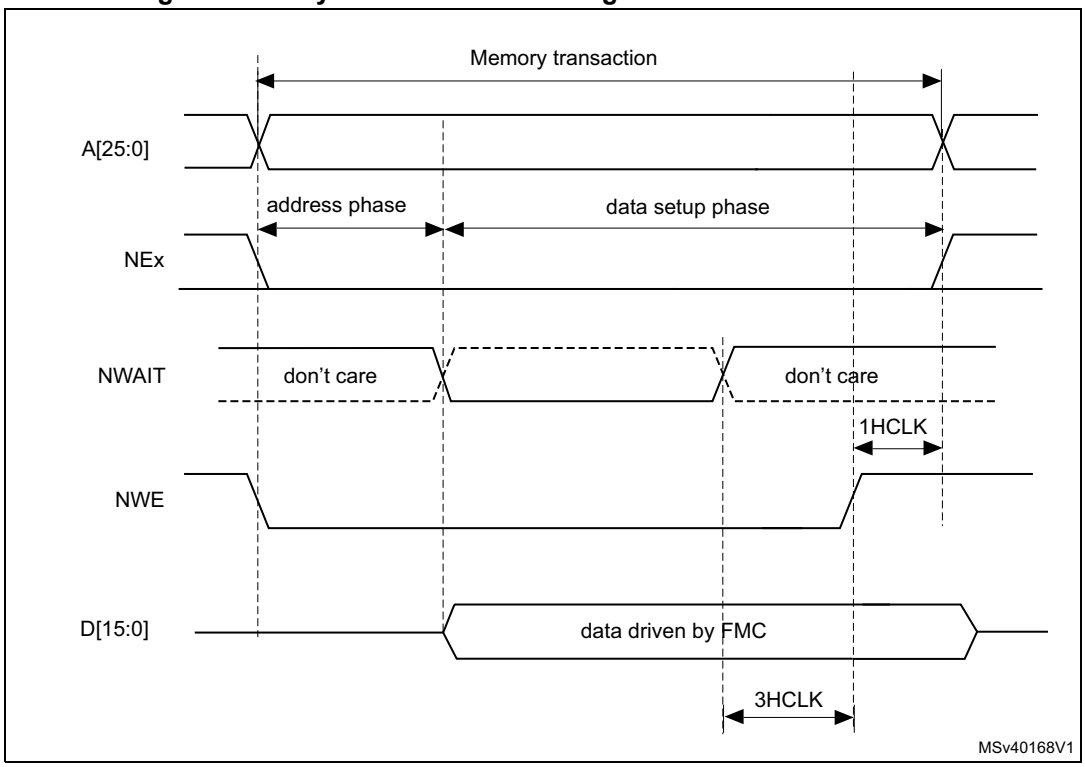
Figure 125 and *Figure 126* show the number of HCLK clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 125. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 126. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

CellularRAM™ (PSRAM) refresh management

The CellularRAM™ does not enable maintaining the chip select signal (NE) low for longer than the t_{CEM} timing specified for the memory device. This timing can be programmed in the FMC_PCSCNTR register. It defines the maximum duration of the NE low pulse in HCLK cycles for asynchronous accesses and FMC_CLK cycles for synchronous accesses

22.6.5 Synchronous transactions

The memory clock, FMC_CLK, is a submultiple of HCLK. It depends on the value of CLKDIV and the MWID/ AHB data size, following the formula given below:

Whatever MWID size: 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

Example:

- If CLKDIV=1, MWID = 16 bits, AHB data size=8 bits, FMC_CLK=HCLK/2.

NOR flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR flash configuration

register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR flash latency and the FMC DATLAT parameter can be either:

- NOR flash latency = (DATLAT + 2) CLK clock cycles
- or NOR flash latency = (DATLAT + 3) CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in Burst mode for synchronous accesses, if for example an AHB single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AHB transfer is 16 bits), or length 2 (if the AHB transfer is 32 bits) and de-assert the chip select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

Cross boundary page for CellularRAM™ 1.5

CellularRAM™ 1.5 does not allow burst access to cross the page boundary. The FMC controller is used to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the chip select and output enable signals valid. It does not consider the data as valid.

In Burst mode, there are two timing configurations for the NOR flash NWAIT signal:

- The flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The flash memory asserts the NWAIT signal during the wait state

The FMC supports both NOR flash wait state configurations, for each chip select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).

Figure 127. Wait configuration waveforms

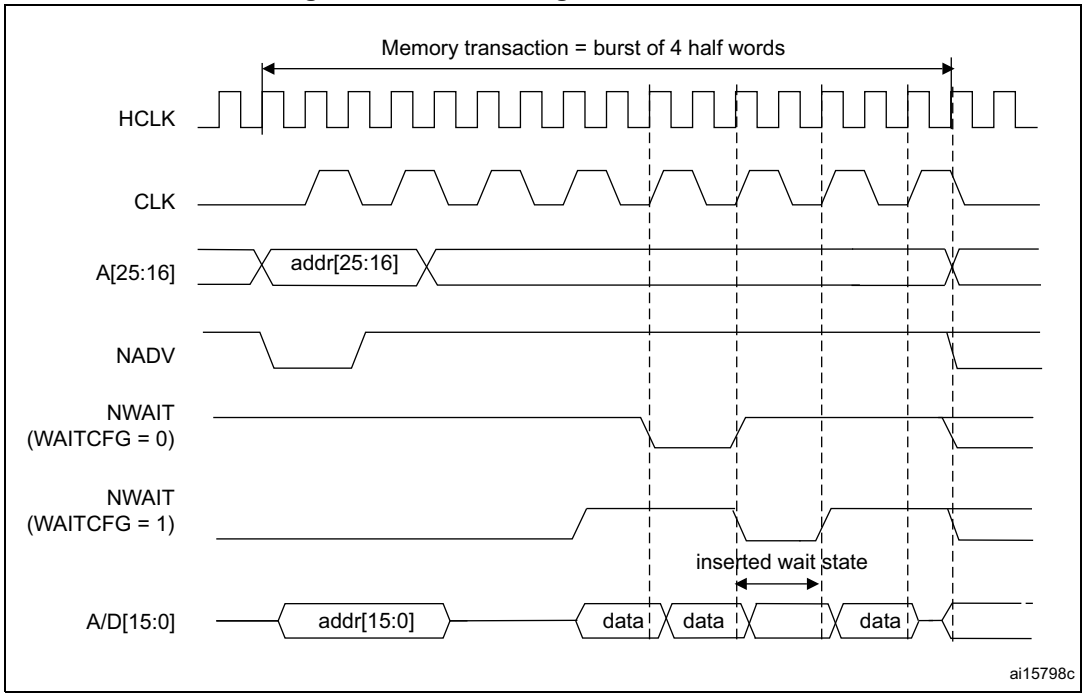
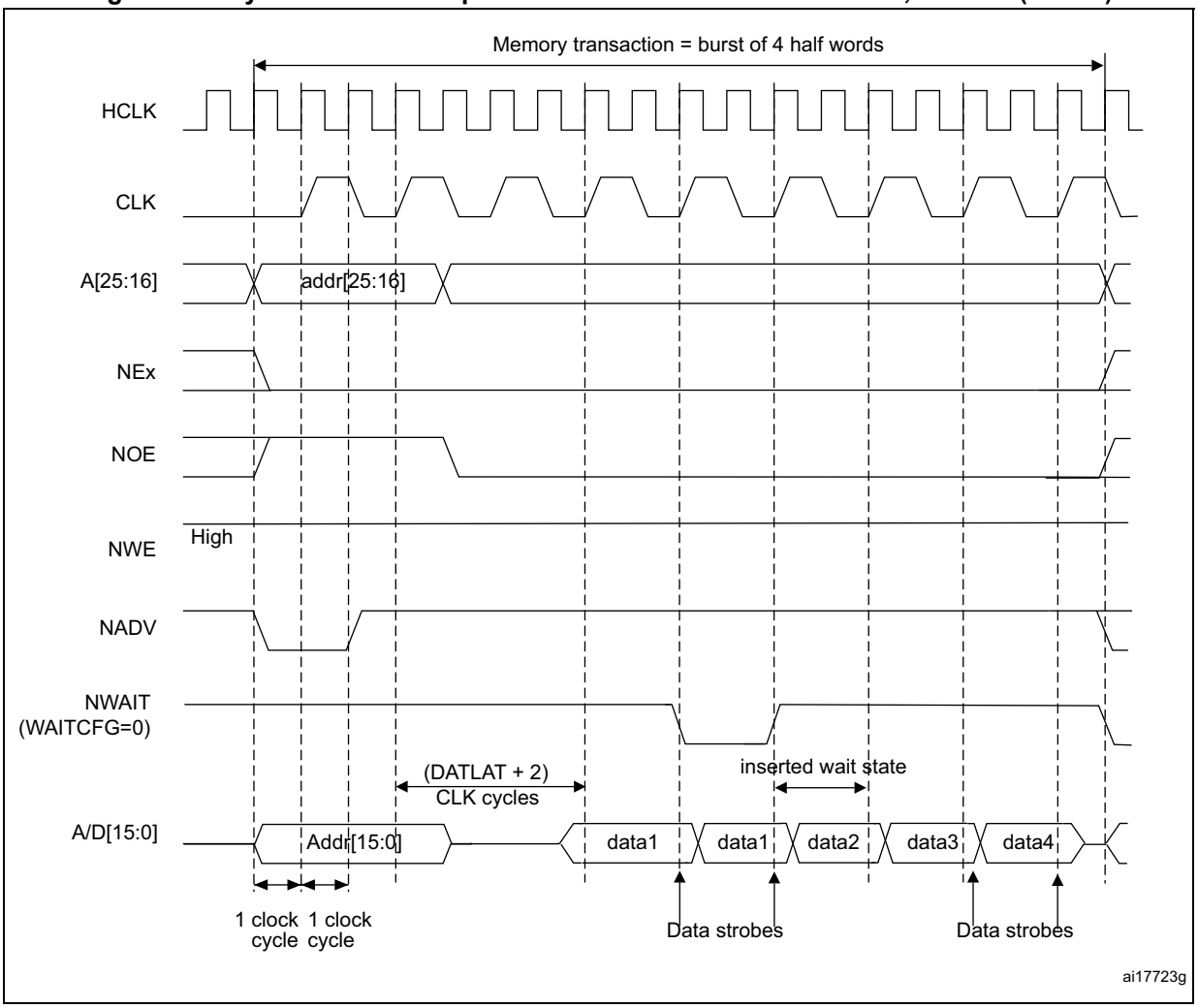


Figure 128. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)



1. Byte lane outputs (NBL are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.

Table 204. FMC_BCRx bitfields (Synchronous multiplexed read mode)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	Don’t care
20	CCLKEN	As needed
19	CBURSTRW	No effect on synchronous read
18:16	CPSIZE	0x0 (no effect in Asynchronous mode)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	To be set to 1 if the memory supports this feature, to be kept at 0 otherwise

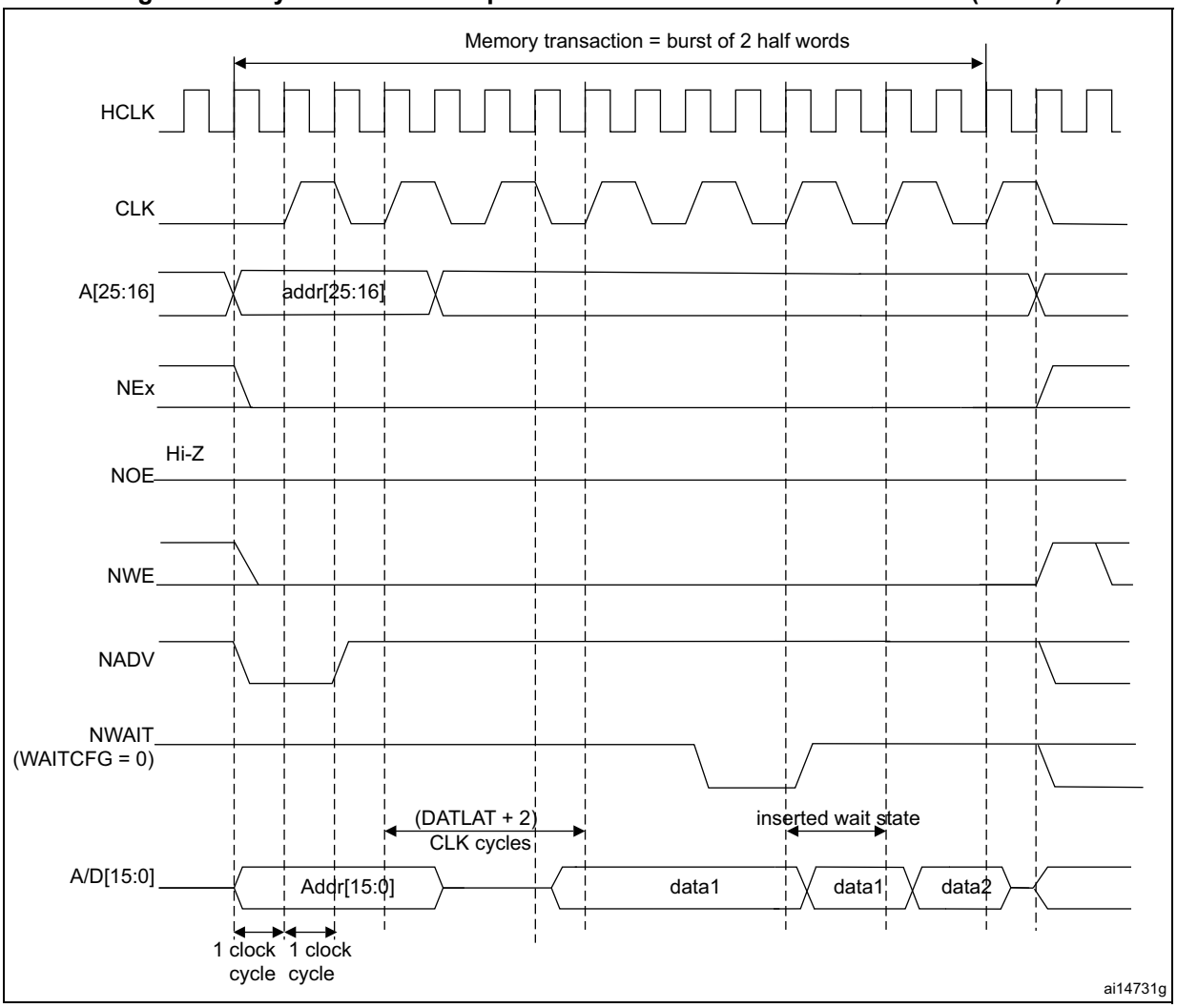
Table 204. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

Bit number	Bit name	Value to set
12	WREN	No effect on synchronous read
11	WAITCFG	To be set according to memory
10	Reserved	0x0
9	WAITPOL	To be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	FACCEN	Set according to memory support (NOR flash memory)
5-4	MWID	As needed
3-2	MTYP	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 205. FMC_BTRx bitfields (Synchronous multiplexed read mode)

Bit number	Bit name	Value to set
31:30	DATAHLD	Don’t care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK ..
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

Figure 129. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 206. FMC_BCRx bitfields (Synchronous multiplexed write mode)

Bit number	Bit name	Value to set
31	FMCEN	0x1
30:24	Reserved	0x000
23:22	NBLSET[1:0]	Don’t care
20	CCLKEN	As needed
19	CBURSTRW	0x1
18:16	CPSIZE	As needed (0x1 for CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0

Table 206. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

Bit number	Bit name	Value to set
13	WAITEN	To be set to 1 if the memory supports this feature, to be kept at 0 otherwise.
12	WREN	0x1
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	no effect on synchronous write
7	Reserved	0x1
6	FACCEN	Set according to memory support
5-4	MWID	As needed
3-2	MTYP	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 207. FMC_BTRx bitfields (Synchronous multiplexed write mode)

Bit number	Bit name	Value to set
31-30	DATAHLD	Don’t care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	0x0 to get CLK = HCLK 0x1 to get CLK = 2 × HCLK
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN HCLK).
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

22.6.6 NOR/PSRAM controller registers

SRAM/NOR-flash chip-select control register for bank x (FMC_BCRx)

Address offset: 0x00 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 30DB, 0x0000 30D2, 0x0000 30D2, 0x0000 30D2

This register contains the control information of each memory bank, used for SRAMs, PSRAM, FRAM and NOR flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		WFDIS	CCLK EN	CBURST RW	CPSIZE[2:0]		
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNC WAIT	EXT MOD	WAIT EN	WREN	WAIT CFG	Res.	WAIT POL	BURST EN	Res.	FACC EN	MWID[1:0]		MTYP[1:0]		MUX EN	MBK EN
rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **FMCEN**: FMC controller enable

This bit enables or disables the FMC controller.

0: Disable the FMC controller

1: Enable the FMC controller

Note: The FMCEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:22 **NBLSET[1:0]:** Byte lane (NBL) setup

These bits configure the NBL setup timing from NBLx low to chip select NEx low.

00: NBL setup time is 0 AHB clock cycle

01: NBL setup time is 1 AHB clock cycle

10: NBL setup time is 2 AHB clock cycles

11: NBL setup time is 3 AHB clock cycles

Bit 21 **WFDIS:** Write FIFO disable

This bit disables the Write FIFO used by the FMC controller.

0: Write FIFO enabled (Default after reset)

1: Write FIFO disabled

Note: The WFDIS bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register.

Bit 20 **CCLKEN:** Continuous clock enable

This bit enables the FMC_CLK clock output to external memory devices.

0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).

1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The CCLKEN bit of the FMC_BCR2..4 registers is don't care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in Synchronous mode to generate the FMC_CLK continuous clock.

Note: If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don't care.

Note: If the Synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 **CBURSTRW:** Write burst enable

For PSRAM (CRAM) operating in Burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.

0: Write operations are always performed in Asynchronous mode.

1: Write operations are performed in Synchronous mode.

Bits 18:16 **CPSIZE[2:0]**: CRAM page size

These are used for CellularRAM™ 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FMC controller splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).

000: No burst split when crossing page boundary (default after reset)

001: 128 bytes

010: 256 bytes

011: 512 bytes

100: 1024 bytes

Others: Reserved, must not be used

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.

0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset).

1: NWAIT signal is taken in to account when running an asynchronous protocol.

Bit 14 **EXTMOD:** Extended mode enable

This bit enables the FMC to program the write timings for non multiplexed asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

0: values inside FMC_BWTR register are not taken into account (default after reset)

1: values inside FMC_BWTR register are taken into account

Note: When the Extended mode is disabled, the FMC can operate in mode 1 or mode 2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).*

Bit 13 **WAITEN:** Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in Synchronous mode.

0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed flash latency period).

1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset).

Bit 12 **WREN:** Write enable bit

This bit indicates whether write operations are enabled/disabled in the bank by the FMC.

0: Write operations are disabled in the bank by the FMC, an AHB error is reported.

1: Write operations are enabled for the bank by the FMC (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in Synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:

0: NWAIT signal is active one data cycle before wait state (default after reset).

1: NWAIT signal is active during wait state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL:** Wait signal polarity bit

Defines the polarity of the wait signal from memory used for either in Synchronous or Asynchronous mode.

0: NWAIT active low (default after reset)

1: NWAIT active high

Bit 8 **BURSTEN:** Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in Burst mode.

0: Burst mode disabled (default after reset). Read accesses are performed in Asynchronous mode.

1: Burst mode enable. Read accesses are performed in Synchronous mode.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FACCEN:** Flash access enable

Enables NOR flash memory access operations.

0: Corresponding NOR flash memory access is disabled.

1: Corresponding NOR flash memory access is enabled (default after reset).

Bits 5:4 **MWID[1:0]:** Memory data bus width

Defines the external memory device width, valid for all type of memories.

00: 8 bits

01: 16 bits (default after reset)

10: reserved

11: reserved

Bits 3:2 **MTYP[1:0]:** Memory type

Defines the type of external memory attached to the corresponding memory bank.

00: SRAM/FRAM (default after reset for Bank 2...4)

01: PSRAM (CRAM) / FRAM

10: NOR flash/OneNAND flash (default after reset for Bank 1)

11: reserved

Bit 1 **MUXEN:** Address/data multiplexing enable bit

When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:

0: Address/data non multiplexed

1: Address/data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory bank enable bit

Enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AHB bus.

0: Corresponding memory bank is disabled.

1: Corresponding memory bank is enabled.

SRAM/NOR-flash chip-select timing register for bank x (FMC_BTRx)

Address offset: 0x04 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR flash memories.If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

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

Bits 31:30 **DATAHLD[1:0]:** Data hold phase duration

These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to *Figure 112* to *Figure 124*), used in asynchronous accesses:

For read accesses

00: DATAHLD phase duration = 0 × HCLK clock cycle (default)

01: DATAHLD phase duration = 1 × HCLK clock cycle

10: DATAHLD phase duration = 2 × HCLK clock cycle

11: DATAHLD phase duration = 3 × HCLK clock cycle

For write accesses

00: DATAHLD phase duration = 1 × HCLK clock cycle (default)

01: DATAHLD phase duration = 2 × HCLK clock cycle

10: DATAHLD phase duration = 3 × HCLK clock cycle

11: DATAHLD phase duration = 4 × HCLK clock cycle

Bits 29:28 **ACCMOD[1:0]:** Access mode

Specifies the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: Access mode A

01: Access mode B

10: Access mode C

11: Access mode D

Bits 27:24 **DATLAT[3:0]:** (see note below bit descriptions): Data latency for synchronous memory
For synchronous access with read/write Burst mode enabled (BURSTEN / CBURSTRW bits set), defines the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:
This timing parameter is not expressed in HCLK periods, but in FMC_CLK periods.
For asynchronous access, this value is don't care.
0000: Data latency of 2 CLK clock cycles for first burst access
1111: Data latency of 17 CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)
Defines the period of FMC_CLK clock output signal, expressed in number of HCLK cycles:
0000: FMC_CLK period= 1x HCLK period
0001: FMC_CLK period = 2 × HCLK periods
0010: FMC_CLK period = 3 × HCLK periods
1111: FMC_CLK period = 16 × HCLK periods (default value after reset)
In asynchronous NOR flash, SRAM or PSRAM accesses, this value is don't care.
Note: Refer to Section 22.6.5: Synchronous transactions for FMC_CLK divider ratio formula)

Bits 19:16 **BUSTURN[3:0]:** Bus turnaround phase duration
These bits are written by software to add a delay at the end of current read or write transaction to next transaction on the same bank.
This delay is used to match the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access (t_{EHQZ}, chip enable high to output Hi-Z). This delay is recommended for mode D and muxed mode. For non-muxed memory, the bus turnaround delay can be set to minimum value.
(BUSTURN + 1)HCLK period ≥ max(t_{EHEL} min, t_{EHQZ} max)
For FRAM memories, the bus turnaround delay must be configured to match the minimum tPC (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read) to match the tPC memory timing. The chip select is toggling between any consecutive accesses.
(BUSTURN + 1)HCLK period ≥ t_{PC} min

0000: BUSTURN phase duration = 1 HCLK clock cycle added
...
1111: BUSTURN phase duration = 16 x HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration
These bits are written by software to define the duration of the data phase (refer to *Figure 112* to *Figure 124*), used in asynchronous accesses:
0000 0000: Reserved
0000 0001: DATAST phase duration = 1 × HCLK clock cycles
0000 0010: DATAST phase duration = 2 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)
For each memory type and access mode data-phase duration, refer to the respective figure (*Figure 112* to *Figure 124*).
Example: Mode 1, write access, DATAST=1: Data-phase duration= DATAST+1 = 2 HCLK clock cycles.
Note: In synchronous accesses, this value is don't care.

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 112* to *Figure 124*), used in mode D or multiplexed accesses:
0000: Reserved
0001: ADDHLD phase duration =1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
...
1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)
For each access mode address-hold phase duration, refer to the respective figure (*Figure 112* to *Figure 124*).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 112* to *Figure 124*), used in SRAMs, ROMs, asynchronous NOR flash and PSRAM:
0000: ADDSET phase duration = 0 × HCLK clock cycle
...
1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)
For each access mode address setup phase duration, refer to the respective figure (*Figure 112* to *Figure 124*).

Note: In synchronous accesses, this value is don't care.
In Muxed mode or mode D, the minimum value for ADDSET is 1.
In mode 1 and PSRAM memory, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to prolong the latency as needed.
With PSRAMs (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.
This method can be used also with the latest generation of synchronous flash memories that issue the NWAIT signal, unlike older flash memories (check the datasheet of the specific flash memory being used).

SRAM/NOR-flash write timing registers x (FMC_BWTRx)

Address offset: 0x104 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRAMs and NOR flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

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

Bits 31:30 **DATAHLD[1:0]:** Data hold phase duration

These bits are written by software to define the duration of the data hold phase in HCLK cycles (refer to *Figure 112* to *Figure 124*), used in asynchronous write accesses:
00: DATAHLD phase duration = 1 × HCLK clock cycle (default)
01: DATAHLD phase duration = 2 × HCLK clock cycle
10: DATAHLD phase duration = 3 × HCLK clock cycle
11: DATAHLD phase duration = 4 × HCLK clock cycle

Bits 29:28 **ACCMOD[1:0]:** Access mode.

Specifies the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.
00: Access mode A
01: Access mode B
10: Access mode C
11: Access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of current write transaction to next transaction on the same bank.
For FRAM memories, the bus turnaround delay must be configured to match the minimum t_{PC} (precharge time) timings. The bus turnaround delay is inserted between any consecutive transactions on the same bank (read/read, write/write, read/write and write/read). The chip select is toggling between any consecutive accesses.
(BUSTURN + 1)HCLK period ≥ tPC min

0000: BUSTURN phase duration = 1 HCLK clock cycle added
...
1111: BUSTURN phase duration = 16 x HCLK clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration.

These bits are written by software to define the duration of the data phase (refer to *Figure 112* to *Figure 124*), used in asynchronous SRAM, PSRAM and NOR flash memory accesses:
0000 0000: Reserved
0000 0001: DATAST phase duration = 1 × HCLK clock cycles
0000 0010: DATAST phase duration = 2 × HCLK clock cycles
...
1111 1111: DATAST phase duration = 255 × HCLK clock cycles (default value after reset)

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration.

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 121* to *Figure 124*), used in asynchronous multiplexed accesses:
0000: Reserved
0001: ADDHLD phase duration = 1 × HCLK clock cycle
0010: ADDHLD phase duration = 2 × HCLK clock cycle
...
1111: ADDHLD phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous NOR flash accesses, this value is not used, the address hold phase is always 1 flash clock period duration.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration.

These bits are written by software to define the duration of the *address setup* phase in HCLK cycles (refer to *Figure 112* to *Figure 124*), used in asynchronous accesses:

0000: ADDSET phase duration = 0 × HCLK clock cycle

...

1111: ADDSET phase duration = 15 × HCLK clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, the address setup phase is always 1 flash clock period duration. In muxed mode, the minimum ADDSET value is 1.

PSRAM chip select counter register (FMC_PCSCNTR)

Address offset: 0x20

Reset value: 0x0000 0000

This register contains the PSRAM chip select counter value for Synchronous and Asynchronous modes. The chip select counter is common to all banks and can be enabled separately on each bank. During PSRAM read or write accesses, this value is loaded into a timer which is decremented while the NE signal is held low. When the timer reaches 0, the PSRAM controller splits the current access, toggles NE to allow PSRAM device refresh, and restarts a new access. The programmed counter value guarantees a maximum NE pulse width (t_{CEM}) as specified for PSRAM devices. The counter is reloaded and starts decrementing each time a new access is started by a transition of NE from high to low.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTB4EN	CNTB3EN	CNTB2EN	CNTB1EN
												rw	rw	rw	rw

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

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CNTB4EN:** Counter Bank 4 enable

This bit enables the chip select counter for PSRAM/NOR Bank 4.

0: Counter disabled for Bank 4

1: Counter enabled for Bank 4

Bit 18 **CNTB3EN:** Counter Bank 3 enable

This bit enables the chip select counter for PSRAM/NOR Bank 3.

0: Counter disabled for Bank 3.

1: Counter enabled for Bank 3

Bit 17 **CNTB2EN:** Counter Bank 2 enable

This bit enables the chip select counter for PSRAM/NOR Bank 2.

0: Counter disabled for Bank 2

1: Counter enabled for Bank 2

Bit 16 **CNTB1EN:** Counter Bank 1 enable

This bit enables the chip select counter for PSRAM/NOR Bank 1.

0: Counter disabled for Bank 1

1: Counter enabled for Bank 1

Bits 15:0 **CSCOUNT[15:0]:** Chip select counter.

This bitfield is used to define the maximum duration of the chip select low, which is obtained by the formula:

CSCOUNT[15:0] * T_{AHB}, where T_{AHB} is the AHB clock period.

For refresh considerations, the PSRAM chip select must not stay low for more than t_{CEM} = ~4 µs.

CSCOUNT[15:0] applies both to asynchronous and synchronous modes.

When CSCOUNT[15:0] = 0x0000, the feature is disabled.

22.7 NAND flash controller

The FMC generates the appropriate signal timings to drive the following types of device:

- 8- and 16-bit NAND flash memories

The NAND bank is configured through dedicated registers (*Section 22.7.7*). The programmable memory parameters include access timings (shown in *Table 208*) and ECC configuration.

Table 208. Programmable NAND flash access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (HCLK) required to set up the address before the command assertion	Read/Write	AHB clock cycle (HCLK)	1	255
Memory wait	Minimum duration (in HCLK clock cycles) of the command assertion	Read/Write	AHB clock cycle (HCLK)	2	255
Memory hold	Number of clock cycles (HCLK) during which the address must be held (as well as the data if a write access is performed) after the command de-assertion	Read/Write	AHB clock cycle (HCLK)	1	254
Memory databus high-Z	Number of clock cycles (HCLK) during which the data bus is kept in high-Z state after a write access has started	Write	AHB clock cycle (HCLK)	1	255

22.7.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND flash memory.

Note: The prefix “N” identifies the signals which are active low.

8-bit NAND flash memory

Table 209. 8-bit NAND flash

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus

Table 209. 8-bit NAND flash (continued)

FMC signal name	I/O	Function
NCE	O	Chip select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND flash memory

Table 210. 16-bit NAND flash

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE	O	Chip select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT	I	NAND flash ready/busy input signal to the FMC

Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

22.7.2 NAND flash supported memories and transactions

Table 211 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND flash controller are shown in gray.

Table 211. Supported memories and transactions

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	-
	Asynchronous	W	8	8	Y	-
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses

Table 211. Supported memories and transactions (continued)

Device	Mode	R/W	AHB data size	Memory data size	Allowed/ not allowed	Comments
NAND 16-bit	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses

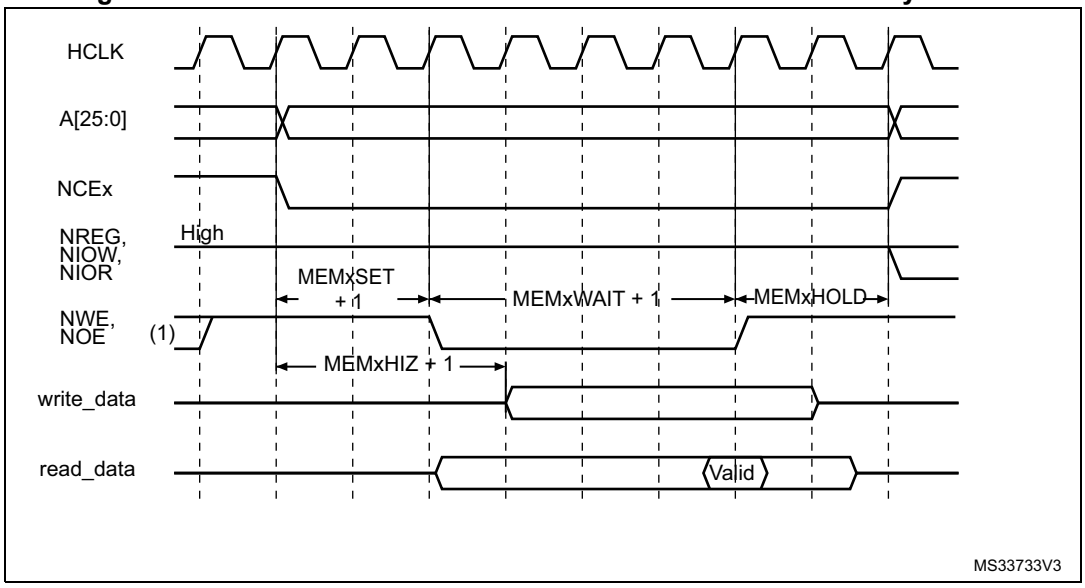
22.7.3 Timing diagrams for NAND flash memory

The NAND flash memory bank is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_ECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

Each timing configuration register contains three parameters used to define number of HCLK cycles for the three phases of any NAND flash access, plus one parameter that defines the timing for starting driving the data bus when a write access is performed. *Figure 130* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 130. NAND flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. For write access, the hold phase delay is (MEMHOLD) HCLK cycles and for read access is (MEMHOLD + 2) HCLK cycles.

22.7.4 NAND flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND flash memory device are driven by address signals from the FMC controller. This means that to send a command or an address to the NAND flash memory, the CPU has to perform a write to a specific address in its memory space.

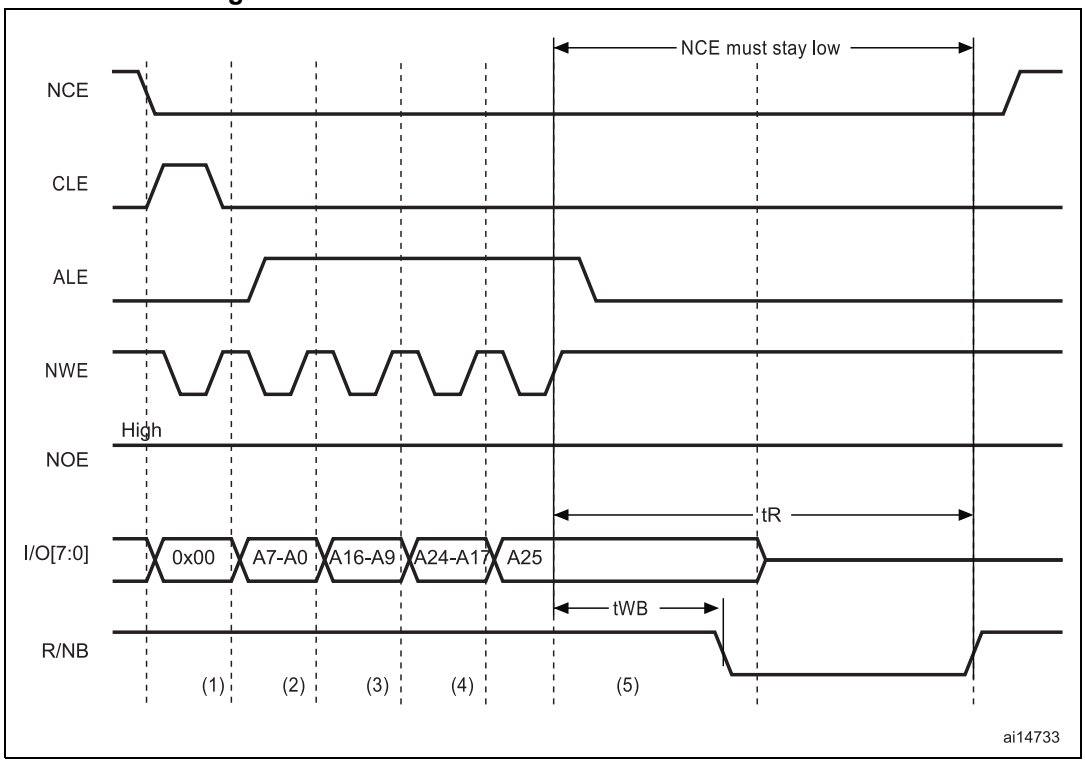
A typical page read operation from the NAND flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 22.7.5: NAND flash prewait functionality*) registers according to the characteristics of the NAND flash memory (PWID bits for the data bus width of the NAND flash, PTYP = 1, PWAITEN = 0 or 1 as needed, see *Section 22.5.2: NAND flash memory address mapping* for timing configuration).
2. The CPU performs a byte write to the common memory space, with data byte equal to one flash command byte (for example 0x00 for Samsung NAND flash devices). The LE input of the NAND flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND flash memories) in the common memory or attribute space. The ALE input of the NAND flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used to implement the prewait functionality needed by some NAND flash memories (see details in *Section 22.7.5: NAND flash prewait functionality*).
4. The controller waits for the NAND flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).
5. The CPU can then perform byte read operations from the common memory space to read the NAND flash page (data field + Spare field) byte by byte.
6. The next NAND flash page can be read without any CPU command or address write operation. This can be done in three different ways:
   - by simply performing the operation described in step 5
   - a new random address can be accessed by restarting the operation at step 3
   - a new command can be sent to the NAND flash device by restarting at step 2

22.7.5 NAND flash prewait functionality

Some NAND flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see *Figure 131*).

Figure 131. Access to non 'CE don't care' NAND-flash



1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7~A0 at address 0x7002 0000.
3. CPU wrote byte A16~A9 at address 0x7002 0000.
4. CPU wrote byte A24~A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x7802 0000: FMC performs a write access using FMC_PATT timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × HCLK = 112 ns > t_{WB} max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND flash memories where NCE is not don't care).

When this functionality is required, it can be ensured by programming the MEMHOLD value to meet the t_{WB} timing. However any CPU read access to the NAND flash memory has a hold delay of (MEMHOLD + 2) HCLK cycles and CPU write access has a hold delay of (MEMHOLD) HCLK cycles inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t_{WB} timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND flash read and write accesses, except when writing the last address byte to the NAND flash device, where the CPU must write to the attribute memory space.

22.7.6 Computation of the error correction code (ECC) in NAND flash memory

The FMC NAND Card controller includes two error correction code computation hardware blocks, one per memory bank. They reduce the host CPU workload when processing the ECC by software.

These two ECC blocks are identical and associated with Bank 2 and Bank 3. As a consequence, no hardware ECC computation is available for memories connected to Bank 4.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1 024, 2 048, 4 096 or 8 192 bytes read or written from/to the NAND flash memory. It is based on the Hamming coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND flash data bus and read/write signals (NCE and NWE) each time the NAND flash memory bank is active.

The ECC operates as follows:

- When accessing NAND flash memory bank 2 or bank 3, the data present on the D[15:0] bus is latched and used for ECC computation.
- When accessing any other address in NAND flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND flash memory by the host CPU, the FMC_ECCR registers must be read to retrieve the computed value. Once read, they must be cleared by resetting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to one in the FMC_PCR registers.

To perform an ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Read the ECC value available in the FMC_ECCR register and store it in a variable.
4. Clear the ECCEN bit and then enable it in the FMC_PCR register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
5. Read the new ECC value available in the FMC_ECCR register.
6. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.

22.7.7 NAND flash controller registers

NAND flash control registers (FMC_PCR)

Address offset: 0x80

Reset value: 0x0000 0018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS[2:0]			TAR3
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				Res.	Res.	ECCEN	PWID[1:0]		PTYP	PBKEN	PWAITEN	Res.
rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:17 **ECCPS[2:0]:** ECC page size
Defines the page size for the extended ECC:
000: 256 bytes
001: 512 bytes
010: 1024 bytes
011: 2048 bytes
100: 4096 bytes
101: 8192 bytes

Bits 16:13 **TAR[3:0]:** ALE to RE delay
Sets time from ALE low to RE low in number of AHB clock cycles (HCLK).
Time is: t_ar = (TAR + SET + 2) × THCLK where THCLK is the HCLK clock period
0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles
Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 12:9 **TCLR[3:0]:** CLE to RE delay
Sets time from CLE low to RE low in number of AHB clock cycles (HCLK).
Time is t_clr = (TCLR + SET + 2) × THCLK where THCLK is the HCLK clock period
0000: 1 HCLK cycle (default)
1111: 16 HCLK cycles
Note: SET is MEMSET or ATTSET according to the addressed space.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **ECCEN:** ECC computation logic enable bit
0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.

Bits 5:4 **PWID[1:0]:** Data bus width
Defines the external memory device width.
00: 8 bits
01: 16 bits (default after reset).
10: reserved.
11: reserved.

Bit 3 **PTYP:** Memory type
Defines the type of device attached to the corresponding memory bank:
0: Reserved, must be kept at reset value
1: NAND flash (default after reset)

Bit 2 **PBKEN:** NAND flash memory bank enable bit
Enables the memory bank. Accessing a disabled memory bank causes an ERROR on AHB bus
0: Corresponding memory bank is disabled (default after reset)
1: Corresponding memory bank is enabled

Bit 1 **PWAITEN:** Wait feature enable bit
Enables the Wait feature for the NAND flash memory bank:
0: disabled
1: enabled

Bit 0 Reserved, must be kept at reset value.

FIFO status and interrupt register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data from the AHB.

This is used to quickly write to the FIFO and free the AHB for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **FEMPT:** FIFO empty
Read-only bit that provides the status of the FIFO
0: FIFO not empty
1: FIFO empty

Bit 5 **IFEN:** Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 4 **ILEN:** Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 3 **IREN:** Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

Bit 2 **IFS:** Interrupt falling edge status
The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred
Note: If this bit is written by software to 1 it is set.

Bit 1 **ILS:** Interrupt high-level status
The flag is set by hardware and reset by software.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IRS:** Interrupt rising edge status

The flag is set by hardware and reset by software.

0: No interrupt rising edge occurred

1: Interrupt rising edge occurred

Note: If this bit is written by software to 1 it is set.

Common memory space timing register (FMC_PMEM)

Address offset: 0x88

Reset value: 0xFCFC FCFC

The FMC_PMEM read/write register contains the timing information for NAND flash memory bank. This information is used to access either the common memory space of the NAND flash for command, address write access and data read/write access.

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

Bits 31:24 **MEMHIZ[7:0]:** Common memory x data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept Hi-Z after the start of a NAND flash write access to common memory space on socket. This is only valid for write transactions:

0000 0000: 1 HCLK cycle

1111 1110: 255 HCLK cycles

1111 1111: reserved.

Bits 23:16 **MEMHOLD[7:0]:** Common memory hold time

Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for NAND flash read or write access to common memory space on socket x:

0000 0000: reserved.

0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access

1111 1110: 254 HCLK cycles for write access / 256 HCLK cycles for read access

1111 1111: reserved.

Bits 15:8 **MEMWAIT[7:0]:** Common memory wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to common memory space on socket. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:

0000 0000: reserved

0000 0001: 2HCLK cycles (+ wait cycle introduced by deasserting NWAIT)

1111 1110: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT)

1111 1111: reserved.

Bits 7:0 **MEMSET[7:0]:** Common memory x setup time

Defines the number of HCLK (+1) clock cycles to set up the address before the command assertion (NWE, NOE), for NAND flash read or write access to common memory space on socket x:
0000 0000: 1 HCLK cycle
1111 1110: 255 HCLK cycles
1111 1111: reserved

Attribute memory space timing register (FMC_PATT)

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FMC_PATT read/write register contains the timing information for NAND flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to *Section 22.7.5: NAND flash prewait functionality*).

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

Bits 31:24 **ATTHIZ[7:0]:** Attribute memory data bus Hi-Z time

Defines the number of HCLK clock cycles during which the data bus is kept in Hi-Z after the start of a NAND flash write access to attribute memory space on socket. Only valid for writ transaction:
0000 0000: 1 HCLK cycle
1111 1110: 255 HCLK cycles
1111 1111: reserved.

Bits 23:16 **ATTHOLD[7:0]:** Attribute memory hold time

Defines the number of HCLK clock cycles for write access and HCLK (+2) clock cycles for read access during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for NAND flash read or write access to attribute memory space on socket:
0000 0000: reserved
0000 0001: 1 HCLK cycle for write access / 3 HCLK cycles for read access
1111 1110: 254 HCLK cycles for write access / 256 HCLK cycles for read access
1111 1111: reserved.

Bits 15:8 **ATTWAIT[7:0]:** Attribute memory wait time

Defines the minimum number of HCLK (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to attribute memory space on socket x. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK:
0000 0000: reserved
0000 0001: 2 HCLK cycles (+ wait cycle introduced by deassertion of NWAIT)
1111 1110: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT)
1111 1111: reserved.

Bits 7:0 **ATTSET[7:0]:** Attribute memory setup time

Defines the number of HCLK (+1) clock cycles to set up address before the command assertion (NWE, NOE), for NAND flash read or write access to attribute memory space on socket:
0000 0000: 1 HCLK cycle
1111 1110: 255 HCLK cycles
1111 1111: reserved.

ECC result registers (FMC_ECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FMC NAND controller. When the CPU reads the data from a NAND flash memory page at the correct address (refer to *Section 22.7.6: Computation of the error correction code (ECC) in NAND flash memory*), the data read/written from/to the NAND flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCR registers), the CPU must read the computed ECC value from the FMC_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECCR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set to 1.

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

Bits 31:0 **ECC[31:0]:** ECC result

This field contains the value computed by the ECC computation logic. *Table 212* describes the contents of these bitfields.

Table 212. ECC result relevant bits

ECCPS[2:0]	Page size in bytes	ECC bits
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

22.8 SDRAM controller

22.8.1 SDRAM controller main features

The main features of the SDRAM controller are the following:

- Two SDRAM banks with independent configuration
- 8-bit, 16-bit data bus width
- 13-bits Address Row, 11-bits Address Column, 4 internal banks: 4x16Mx16bit (128 MB), 4x16Mx8bit (64 MB)
- Word, half-word, byte access
- Automatic row and bank boundary management
- Multibank ping-pong access
- Programmable timing parameters
- Automatic Refresh operation with programmable Refresh rate
- Self-refresh mode
- Power-down mode
- SDRAM power-up initialization by software
- CAS latency of 1,2,3
- Cacheable Read FIFO with depth of 6 lines x32-bit (6 x14-bit address tag)

22.8.2 SDRAM External memory interface signals

At startup, the SDRAM I/O pins used to interface the FMC SDRAM controller with the external SDRAM devices must configured by the user application. The SDRAM controller I/O pins which are not used by the application, can be used for other purposes.

Table 213. SDRAM signals

SDRAM signal	I/O type	Description	Alternate function
SDCLK	O	SDRAM clock	-
SDCKE[1:0]	O	SDCKE0: SDRAM Bank 1 Clock Enable SDCKE1: SDRAM Bank 2 Clock Enable	-
SDNE[1:0]	O	SDNE0: SDRAM Bank 1 Chip Enable SDNE1: SDRAM Bank 2 Chip Enable	-
A[12:0]	O	Address	FMC_A[12:0]
D[15:0]	I/O	Bidirectional data bus	FMC_D[15:0]
BA[1:0]	O	Bank Address	FMC_A[15:14]
NRAS	O	Row Address Strobe	-
NCAS	O	Column Address Strobe	-
SDNWE	O	Write Enable	-
NBL[1:0]	O	Output Byte Mask for write accesses (memory signal name: DQM[1:0]	FMC_NBL[1:0]

22.8.3 SDRAM controller functional description

All SDRAM controller outputs (signals, address and data) change on the falling edge of the memory clock (FMC_SDCLK).

SDRAM initialization

The initialization sequence is managed by software. If the two banks are used, the initialization sequence must be generated simultaneously to Bank 1and Bank 2 by setting the Target Bank bits CTB1 and CTB2 in the FMC_SDCMR register:

1. Program the memory device features into the FMC_SDCRx register. The SDRAM clock frequency, RBURST and RPIPE must be programmed in the FMC_SDCR1 register.
2. Program the memory device timing into the FMC_SDTRx register. The TRP and TRC timings must be programmed in the FMC_SDTR1 register.
3. Set MODE bits to ‘001’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to start delivering the clock to the memory (SDCKE is driven high).
4. Wait during the prescribed delay period. Typical delay is around 100 µs (refer to the SDRAM datasheet for the required delay after power-up).
5. Set MODE bits to ‘010’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Precharge All” command.
6. Set MODE bits to ‘011’, and configure the Target Bank bits (CTB1 and/or CTB2) as well as the number of consecutive Auto-refresh commands (NRFS) in the FMC_SDCMR register. Refer to the SDRAM datasheet for the number of Auto-refresh commands that should be issued. Typical number is 8.
7. Configure the MRD field according to the SDRAM device, set the MODE bits to '100', and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a "Load Mode Register" command in order to program the SDRAM device. In particular:
   a) the CAS latency must be selected following configured value in FMC_SDCR1/2 registers
   b) the Burst Length (BL) of 1 must be selected by configuring the M[2:0] bits to 000 in the mode register. Refer to SDRAM device datasheet.

   If the Mode Register is not the same for both SDRAM banks, this step has to be repeated twice, once for each bank, and the Target Bank bits set accordingly.
8. Program the refresh rate in the FMC_SDRTR register

   The refresh rate corresponds to the delay between refresh cycles. Its value must be adapted to SDRAM devices.
9. For mobile SDRAM devices, to program the extended mode register it should be done once the SDRAM device is initialized: First, a dummy read access should be performed while BA1=1 and BA=0 (refer to SDRAM address mapping section for BA[1:0] address mapping) in order to select the extended mode register instead of the load mode register and then program the needed value.

At this stage the SDRAM device is ready to accept commands. If a system reset occurs during an ongoing SDRAM access, the data bus might still be driven by the SDRAM device. Therefor the SDRAM device must be first reinitialized after reset before issuing any new access by the NOR flash/PSRAM/SRAM or NAND flash controller.

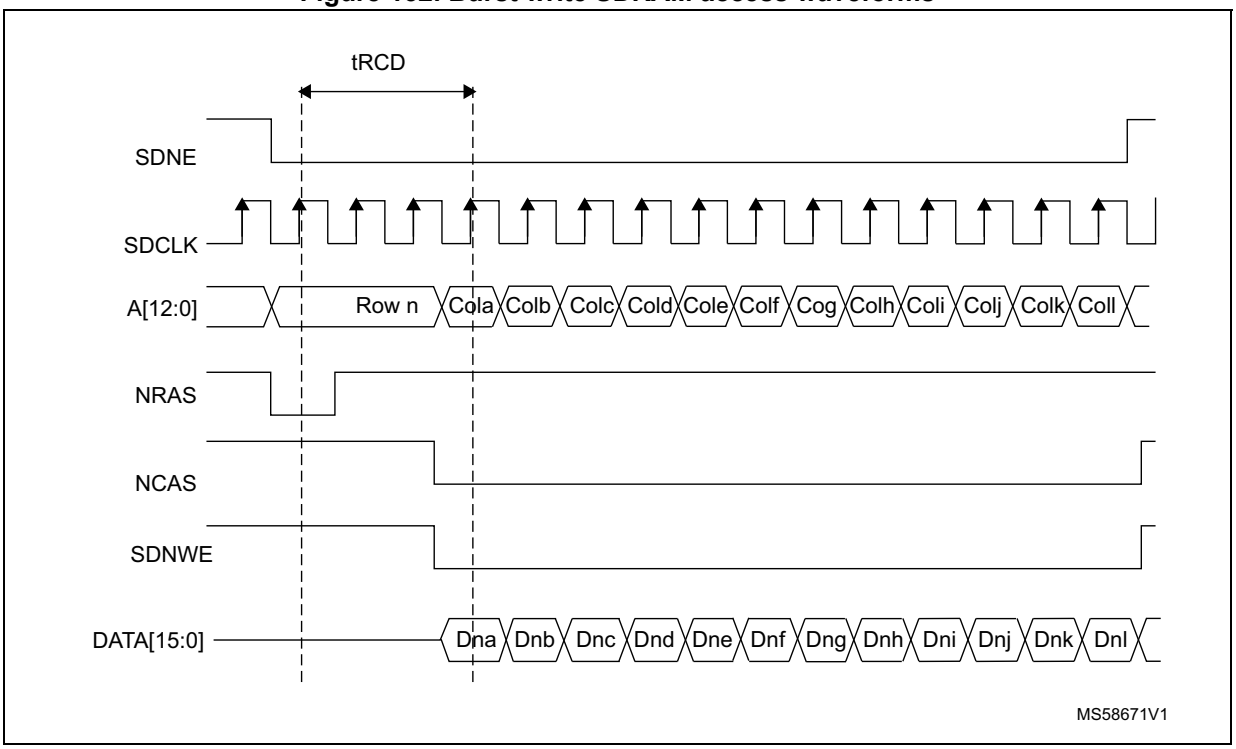
Note: If two SDRAM devices are connected to the FMC, all the accesses performed at the same time to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for SDRAM Bank 1 (TMRD andTRAS timings) in the FMC_SDTR1 register.

SDRAM controller write cycle

The SDRAM controller accepts single and burst write requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row for each bank to be able to perform consecutive write accesses to different banks (Multibank ping-pong access).

Before performing any write access, the SDRAM bank write protection must be disabled by clearing the WP bit in the FMC_SDCRx register.

Figure 132. Burst write SDRAM access waveforms



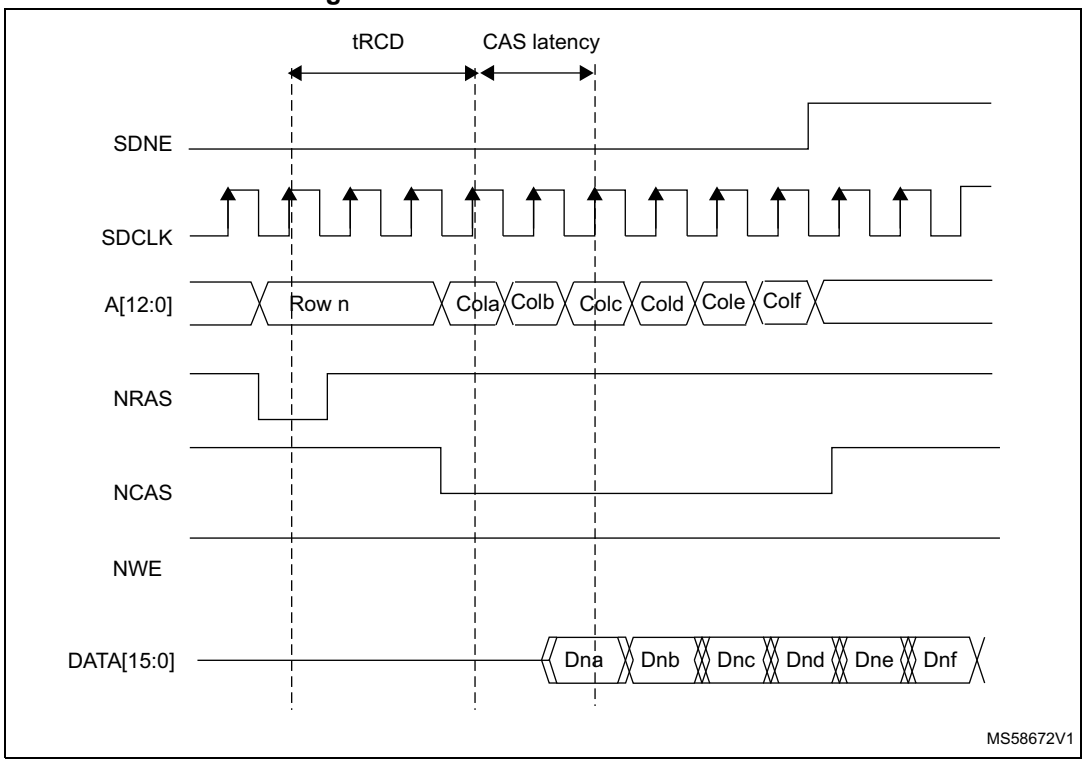
The SDRAM controller always checks the next access.

- If the next access is in the same row or in another active row, the write operation is carried out,
- if the next access targets another row (not active), the SDRAM controller generates a precharge command, activates the new row and initiates a write command.

SDRAM controller read cycle

The SDRAM controller accepts single and burst read requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row in each bank to be able to perform consecutive read accesses in different banks (Multibank ping-pong access).

Figure 133. Burst read SDRAM access



The FMC SDRAM controller features a Cacheable read FIFO (6 lines x 32 bits). It is used to store data read in advance during the CAS latency period and the RPIPE delay following the below formula. The RBURST bit must be set in the FMC_SDCR1 register to anticipate the next read access.

Number for anticipated data = CAS latency + 1 + (RPIPE delay)/2

Examples:

- CAS latency = 3, RPIPE delay = 0: Four data (not committed) are stored in the FIFO.
- CAS latency = 3, RPIPE delay = 2: Five data (not committed) are stored in the FIFO.

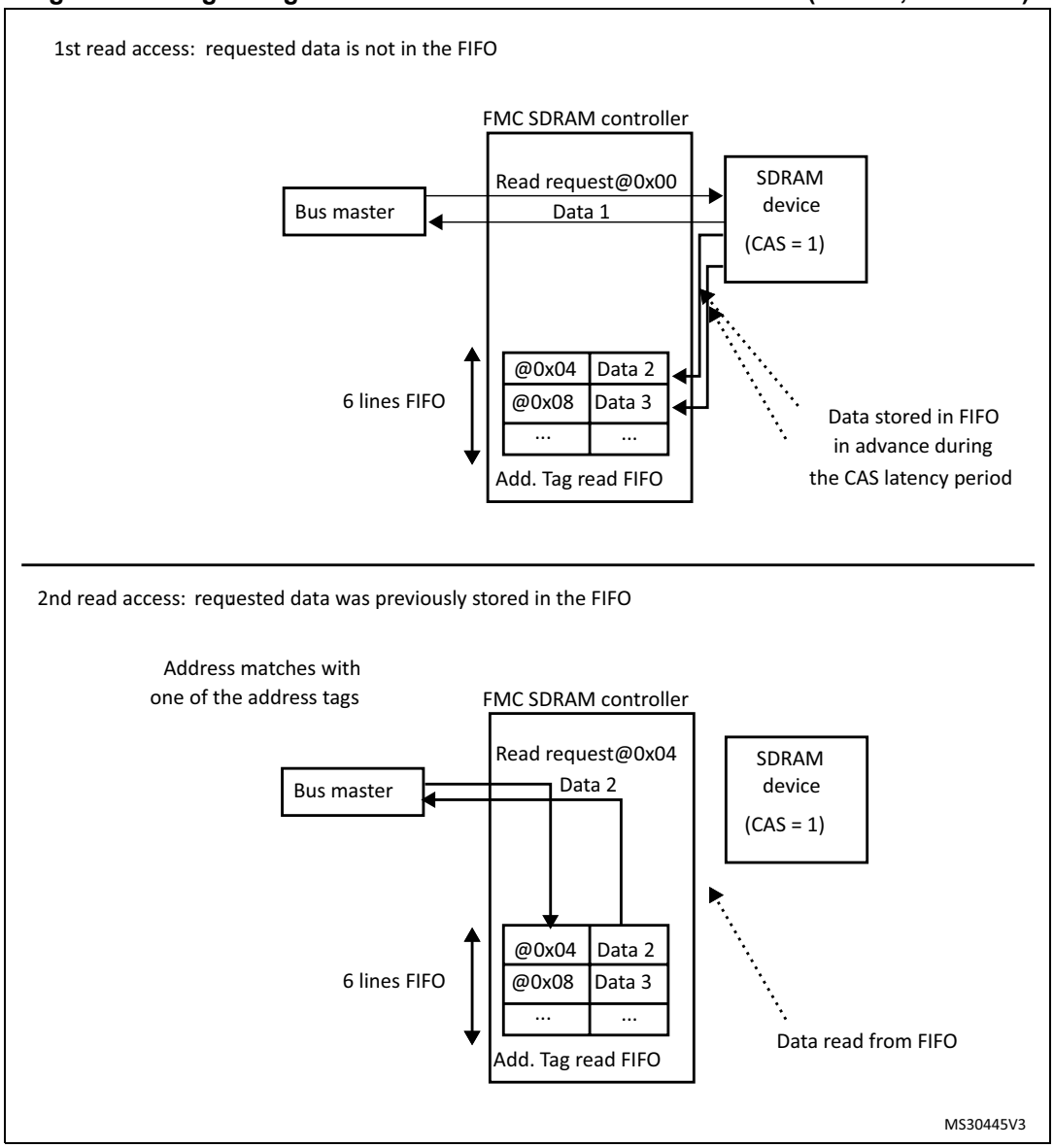
The read FIFO features a 14-bit address tag to each line to identify its content: 11 bits for the column address, 2 bits to select the internal bank and the active row, and 1 bit to select the SDRAM device

When the end of the row is reached in advance during an AHB burst read, the data read in advance (not committed) are not stored in the read FIFO. For single read access, data are correctly stored in the FIFO.

Each time a read request occurs, the SDRAM controller checks:

- If the address matches one of the address tags, data are directly read from the FIFO and the corresponding address tag/ line content is cleared and the remaining data in the FIFO are compacted to avoid empty lines.
- Otherwise, a new read command is issued to the memory and the FIFO is updated with new data. If the FIFO is full, the older data are lost.

Figure 134. Logic diagram of Read access with RBURST bit set (CAS=1, RPIPE=0)



During a write access or a Precharge command, the read FIFO is flushed and ready to be filled with new data.

After the first read request, if the current access was not performed to a row boundary, the SDRAM controller anticipates the next read access during the CAS latency period and the RPIPE delay (if configured). This is done by incrementing the memory address. The following condition must be met:

- RBURST control bit should be set to ‘1’ in the FMC_SDCR1 register.

The address management depends on the next AHB request:

- Next AHB request is sequential (AHB Burst)

  In this case, the SDRAM controller increments the address.
- Next AHB request is not sequential
  - If the new read request targets the same row or another active row, the new address is passed to the memory and the master is stalled for the CAS latency period, waiting for the new data from memory.
  - If the new read request does not target an active row, the SDRAM controller generates a Precharge command, activates the new row, and initiates a read command.

If the RURST is reset, the read FIFO is not used.

Row and bank boundary management

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, the SDRAM controller executes the following operations:

1. Precharge of the active row,
2. Activation of the new row
3. Start of a read/write command.

At a row boundary, the automatic activation of the next row is supported for all columns and data bus width configurations.

If necessary, the SDRAM controller inserts additional clock cycles between the following commands:

- Between Precharge and Active commands to match TRP parameter (only if the next access is in a different row in the same bank),
- Between Active and Read commands to match the TRCD parameter.

These parameters are defined into the FMC_SDTRx register.

Refer to *Figure 132* and *Figure 133* for read and burst write access crossing a row boundary.

Figure 135. Read access crossing row boundary

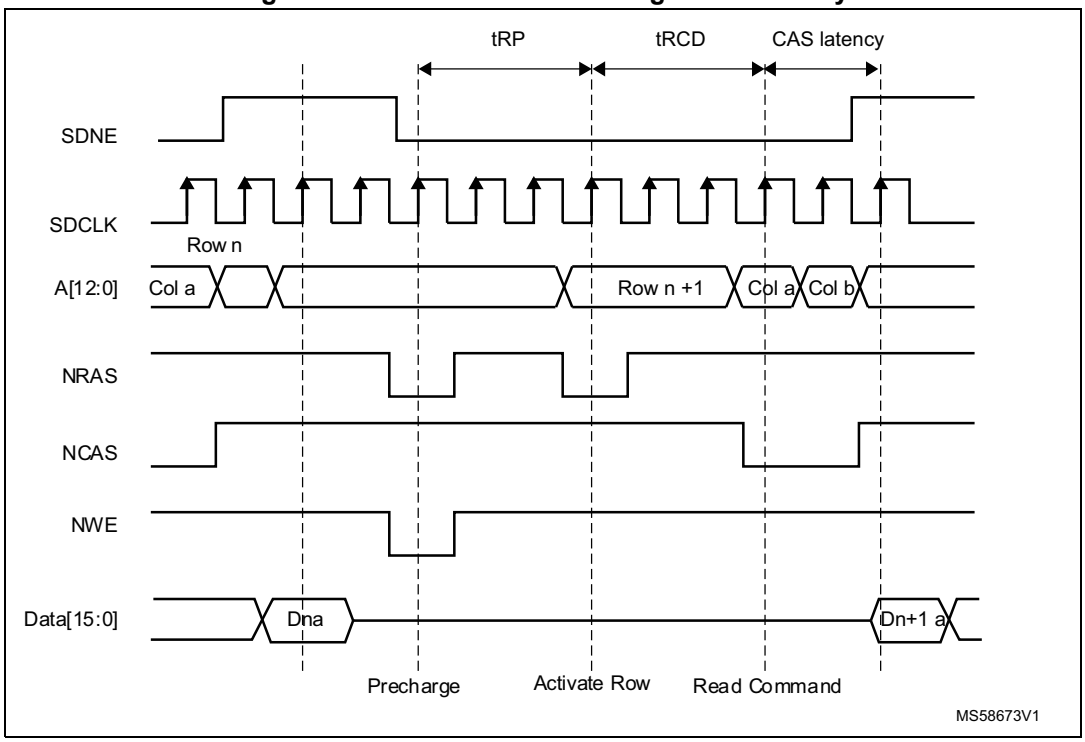
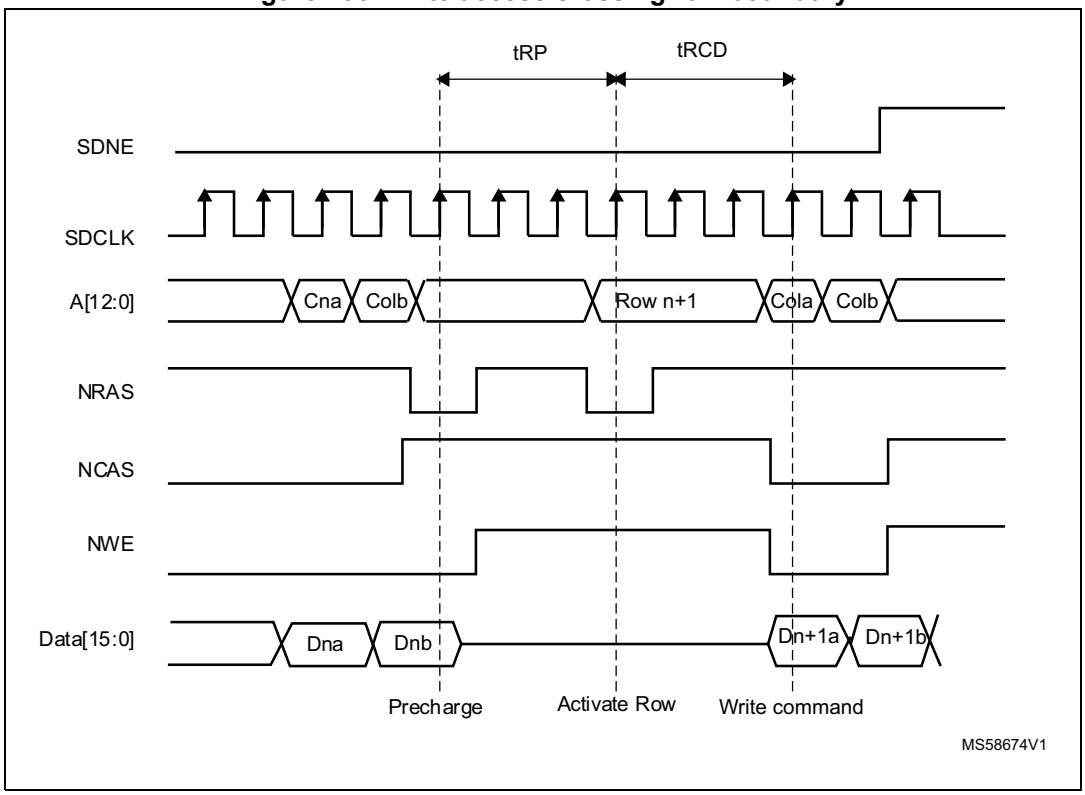


Figure 136. Write access crossing row boundary



If the next access is sequential and the current access crosses a bank boundary, the SDRAM controller activates the first row in the next bank and initiates a new read/write command. Two cases are possible:

- If the current bank is not the last one, the active row in the new bank must be precharged.At a bank boundary, the automatic activation of the next row is supported for all rows/columns and data bus width configuration.
- If the current bank is the last one and the selected SDRAM device is connected to Bank 1, the automatic activation of the next row in device connected to SDRAM Bank2 is not supported. A PALL software command must be issused on Bank1 before any any access on Bank2.

SDRAM controller refresh cycle

The Auto-refresh command is used to refresh the SDRAM device content. The SDRAM controller periodically issues auto-refresh commands. An internal counter is loaded with the COUNT value in the register FMC_SDRTR. This value defines the number of memory clock cycles between the refresh cycles (refresh rate). When this counter reaches zero, an internal pulse is generated.

If a memory access is ongoing, the auto-refresh request is delayed. However, if the memory access and the auto-refresh requests are generated simultaneously, the auto-refresh request takes precedence.

If the memory access occurs during an auto-refresh operation, the request is buffered and processed when the auto-refresh is complete.

If a new auto-refresh request occurs while the previous one was not served, the RE (Refresh Error) bit is set in the Status register. An Interrupt is generated if it has been enabled (REIE = ‘1’).

If SDRAM lines are not in idle state (not all row are closed), the SDRAM controller generates a PALL (Precharge ALL) command before the auto-refresh.

If the Auto-refresh command is generated by the FMC_SDCMR Command Mode register (Mode bits = ‘011’), a PALL command (Mode bits =’ 010’) must be issued first.

22.8.4 Low-power modes

Two low-power modes are available:

- Self-refresh mode

  The auto-refresh cycles are performed by the SDRAM device itself to retain data without external clocking.
- Power-down mode

  The auto-refresh cycles are performed by the SDRAM controller.

Self-refresh mode

This mode is selected by setting the MODE bits to ‘101’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

The SDRAM clock stops running after a TRAS delay and the internal refresh timer stops counting only if one of the following conditions is met:

- A Self-refresh command is issued to both devices
- One of the devices is not activated (SDRAM bank is not initialized).

Before entering Self-Refresh mode, the SDRAM controller automatically issues a PALL command.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Self-refresh mode and the BUSY status flag remains set.

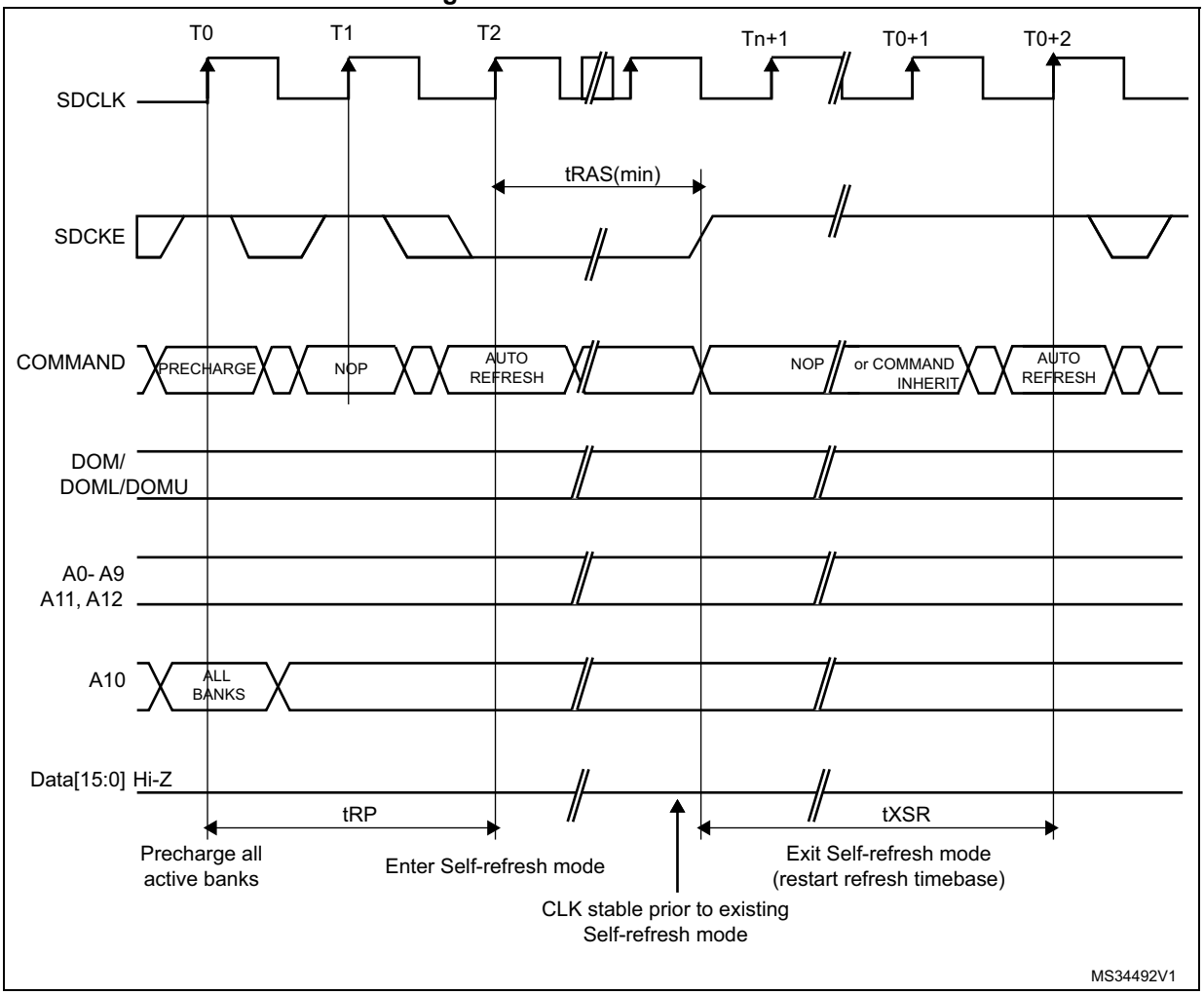
In Self-refresh mode, all SDRAM device inputs become don't care except for SDCKE which remains low.

The SDRAM device must remain in Self-refresh mode for a minimum period of time of TRAS and can remain in Self-refresh mode for an indefinite period beyond that. To guarantee this minimum period, the BUSY status flag remains high after the Self-refresh activation during a TRAS delay.

As soon as an SDRAM device is selected, the SDRAM controller generates a sequence of commands to exit from Self-refresh mode. After the memory access, the selected device remains in Normal mode.

To exit from Self-refresh, the MODE bits must be set to '000' (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.

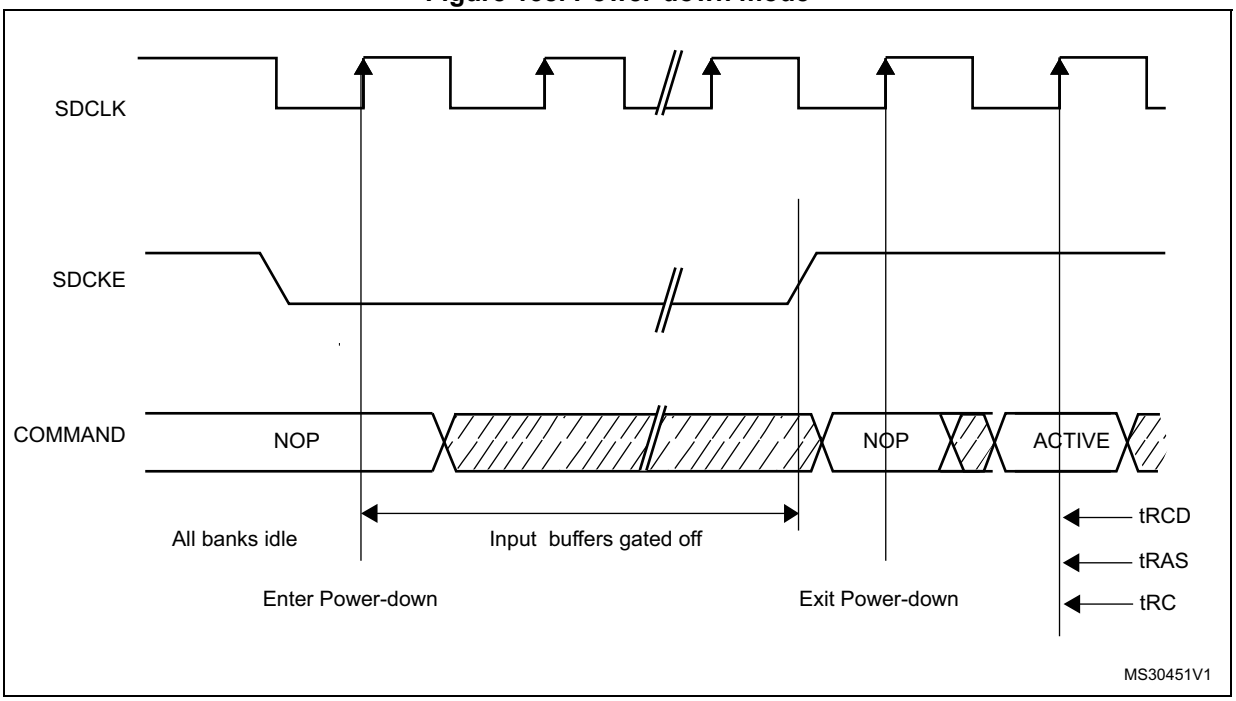
Figure 137. Self-refresh mode



Power-down mode

This mode is selected by setting the MODE bits to ‘110’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

Figure 138. Power-down mode



If the Write data FIFO is not empty, all data are sent to the memory before activating the Power-down mode.

As soon as an SDRAM device is selected, the SDRAM controller exits from the Power-down mode. After the memory access, the selected SDRAM device remains in Normal mode.

During Power-down mode, all SDRAM device input and output buffers are deactivated except for the SDCKE which remains low.

The SDRAM device cannot remain in Power-down mode longer than the refresh period and cannot perform the Auto-refresh cycles by itself. Therefore, the SDRAM controller carries out the refresh operation by executing the operations below:

1. Exit from Power-down mode and drive the SDCKE high
2. Generate the PALL command only if a row was active during Power-down mode
3. Generate the auto-refresh command
4. Drive SDCKE low again to return to Power-down mode.

To exit from Power-down mode, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.

22.8.5 SDRAM controller registers

SDRAM control register x (FMC_SDCRx)

Address offset: 0x140+ 0x4* (x -1), (x = 1,2)

Reset value: 0x0000 02D0

This register contains the control parameters for each SDRAM memory bank

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RPIPE[1:0]		RBURST	SDCLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:13 **RPIPE[1:0]:** Read pipe

These bits define the delay, in clock cycles, for reading data after CAS latency.

00: No clock cycle delay

01: One clock cycle delay

10: Two clock cycle delay

11: reserved.

Note: The corresponding bits in the FMC_SDCR2 register is read only.

Bit 12 **RBURST:** Burst read

This bit enables Burst read mode. The SDRAM controller anticipates the next read commands during the CAS latency and stores data in the Read FIFO.

0: single read requests are not managed as bursts

1: single read requests are always managed as bursts

Note: The corresponding bit in the FMC_SDCR2 register is don't care.

Bits 11:10 **SDCLK[1:0]:** SDRAM clock configuration

These bits define the SDRAM clock period for both SDRAM banks and allow disabling the clock before changing the frequency. In this case the SDRAM must be re-initialized.

00: SDCLK clock disabled

01: SDCLK period = 1x HCLK periods

10: SDCLK period = 2 x HCLK periods

11: SDCLK period = 3 x HCLK periods

Note: The corresponding bits in the FMC_SDCR2 register are don't care.

Bit 9 **WP:** Write protection

This bit enables write mode access to the SDRAM bank.

0: Write accesses allowed

1: Write accesses ignored

Bits 8:7 **CAS[1:0]:** CAS Latency

This bits sets the SDRAM CAS latency in number of memory clock cycles

00: reserved.

01: 1 cycle

10: 2 cycles

11: 3 cycles

Bit 6 **NB:** Number of internal banks
This bit sets the number of internal banks.
0: Two internal Banks
1: Four internal Banks

Bits 5:4 **MWID[1:0]:** Memory data bus width.
These bits define the memory device width.
00: 8 bits
01: 16 bits
10: reserved
11: reserved.

Bits 3:2 **NR[1:0]:** Number of row address bits
These bits define the number of bits of a row address.
00: 11 bit
01: 12 bits
10: 13 bits
11: reserved.

Bits 1:0 **NC[1:0]:** Number of column address bits
These bits define the number of bits of a column address.
00: 8 bits
01: 9 bits
10: 10 bits
11: 11 bits.

Note: Before modifying the RBURST or RPIPE settings or disabling the SDCLK clock, the user must first send a PALL command to make sure ongoing operations are complete.

SDRAM timing register x (FMC_SDTRx)

Address offset: 0x148 + 0x4 * (x - 1), (x = 1,2)

Reset value: 0x0FFF FFFF

This register contains the timing parameters of each SDRAM bank

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TRCD[3:0]:** Row to column delay
These bits define the delay, tRCD, from the Activate command to a Read/Write command in number of memory clock cycles.
0000: 2 cycles
0001: 3 cycles
....
1111: 17 cycles
Note: tRCD delay is replaced by (TWR[3:0]+2) cycles when TWR[3:0] > TRCD[3:0].

Bits 23:20 **TRP[3:0]:** Row precharge delay

These bits define the delay, tRP, between a Precharge command and another command in number of memory clock cycles. The TRP timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP must be programmed with the timing of the slowest device.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: The corresponding bits in the FMC_SDTR2 register are don't care.

Bits 19:16 **TWR[3:0]:** Recovery delay

These bits define the delay, tWR, between a Write and a Precharge command in number of memory clock cycles.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: TWR must be programmed to match the write recovery time (t_{WR}) defined in the SDRAM datasheet, and to guarantee that:

TWR ≥ TRAS - TRCD and TWR ≥TRC - TRCD - TRP

Example: TRAS= 4 cycles, TRCD= 2 cycles. So, TWR >= 2 cycles. TWR must be programmed to 0x1.

If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TWR timing corresponding to the slowest SDRAM device.

If only one SDRAM device is used, the TWR timing must be kept at reset value (0xF) for the not used bank.

Bits 15:12 **TRC[3:0]:** Row cycle delay

These bits define the delay, tRC, between the Refresh command and the Activate command, as well as the delay between two consecutive Refresh commands. It is expressed in number of memory clock cycles. The TRC timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRC must be programmed with the timings of the slowest device.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: TRC must match the TRC and TRFC (Auto Refresh period) timings defined in the SDRAM device datasheet.

Note: The corresponding bits in the FMC_SDTR2 register are don't care.

Bits 11:8 **TRAS[3:0]:** Self refresh time

These bits define the minimum Self-refresh period in number of memory clock cycles.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Bits 7:4 **TXSR[3:0]:** Exit Self-refresh delay

These bits define the delay from releasing the Self-refresh command to issuing the Activate command in number of memory clock cycles.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TXSR timing corresponding to the slowest SDRAM device.

Bits 3:0 **TMRD[3:0]:** Load Mode Register to Active

These bits define the delay, tXSR, between a Load Mode Register command and an Active or Refresh command in number of memory clock cycles.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: If two SDRAM devices are connected, all the accesses performed simultaneously to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.

The TRP and TRC timings are only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP and TRC timings must be programmed with the timings of the slowest device.

SDRAM Command Mode register (FMC_SDCMR)

Address offset: 0x150

Reset value: 0x0000 0000

This register contains the command issued when the SDRAM device is accessed. This register is used to initialize the SDRAM device, and to activate the Self-refresh and the Power-down modes. As soon as the MODE field is written, the command is issued only to one or to both SDRAM banks according to CTB1 and CTB2 command bits. This register is the same for both SDRAM banks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD[12:7]					
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRD[6:0]							NRFS[3:0]				CTB1	CTB2	MODE[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w	w	w	w

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

Bits 21:9 **MRD[12:0]:** Mode Register definition

This 13-bit field defines the SDRAM Mode Register content. The Mode Register is programmed using the Load Mode Register command.

Bits 8:5 **NRFS[3:0]:** Number of Auto-refresh
These bits define the number of consecutive Auto-refresh commands issued when MODE = '011'.
0000: 1 Auto-refresh cycle
0001: 2 Auto-refresh cycles
....
1110: 15 Auto-refresh cycles
1111: 16 Auto-refresh cycles

Bit 4 **CTB1:** Command Target Bank 1
This bit indicates whether the command is issued to SDRAM Bank 1 or not.
0: Command not issued to SDRAM Bank 1
1: Command issued to SDRAM Bank 1

Bit 3 **CTB2:** Command Target Bank 2
This bit indicates whether the command is issued to SDRAM Bank 2 or not.
0: Command not issued to SDRAM Bank 2
1: Command issued to SDRAM Bank 2

Bits 2:0 **MODE[2:0]:** Command mode
These bits define the command issued to the SDRAM device.
000: Normal Mode
001: Clock Configuration Enable
010: PALL ("All Bank Precharge") command
011: Auto-refresh command
100: Load Mode Register
101: Self-refresh command
110: Power-down command
111: Reserved

Note: When a command is issued, at least one Command Target Bank bit (CTB1 or CTB2) must be set otherwise the command is ignored.

Note: If two SDRAM banks are used, the Auto-refresh and PALL command must be issued simultaneously to the two devices with CTB1 and CTB2 bits set otherwise the command is ignored.

Note: If only one SDRAM bank is used and a command is issued with it's associated CTB bit set, the other CTB bit of the the unused bank must be kept to 0.

SDRAM refresh timer register (FMC_SDRTR)

Address offset:0x154

Reset value: 0x0000 0000

This register sets the refresh rate in number of SDCLK clock cycles between the refresh cycles by configuring the Refresh Timer Count value.

$$\text{Refresh rate} = (\text{COUNT} + 1) \times \text{SDRAM clock frequency}$$

$$\text{COUNT} = (\text{SDRAM refresh period} / \text{Number of rows}) - 20$$

Example

$$\text{Refresh rate} = 64\ \text{ms} / (8196\text{rows}) = 7.81\mu\text{s}$$

where 64 ms is the SDRAM refresh period.

$$7.81\mu s \times 60MHz = 468.6$$

The refresh rate must be increased by 20 SDRAM clock cycles (as in the above example) to obtain a safe margin if an internal refresh request occurs when a read request has been accepted. It corresponds to a COUNT value of ‘0000111000000’ (448).

This 13-bit field is loaded into a timer which is decremented using the SDRAM clock. This timer generates a refresh pulse when zero is reached. The COUNT value must be set at least to 41 SDRAM clock cycles.

As soon as the FMC_SDRTR register is programmed, the timer starts counting. If the value programmed in the register is ’0’, no refresh is carried out. This register must not be reprogrammed after the initialization procedure to avoid modifying the refresh rate.

Each time a refresh pulse is generated, this 13-bit COUNT field is reloaded into the counter.

If a memory access is in progress, the Auto-refresh request is delayed. However, if the memory access and Auto-refresh requests are generated simultaneously, the Auto-refresh takes precedence. If the memory access occurs during a refresh operation, the request is buffered to be processed when the refresh is complete.

This register is common to SDRAM bank 1 and bank 2.

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **REIE:** RES Interrupt Enable
0: Interrupt is disabled
1: An Interrupt is generated if RE = 1

Bits 13:1 **COUNT[12:0]:** Refresh Timer Count
This 13-bit field defines the refresh rate of the SDRAM device. It is expressed in number of memory clock cycles. It must be set at least to 41 SDRAM clock cycles (0x29).
Refresh rate = (COUNT + 1) x SDRAM frequency clock
COUNT = (SDRAM refresh period / Number of rows) - 20

Bit 0 **CRE:** Clear Refresh error flag
This bit is used to clear the Refresh Error Flag (RE) in the Status Register.
0: no effect
1: Refresh Error flag is cleared

Note: The programmed COUNT value must not be equal to the sum of the following timings: TWR+TRP+TRC+TRCD+4 memory clock cycles.

SDRAM status register (FMC_SDSR)

Address offset: 0x158

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	MODES2[1:0]		MODES1[1:0]		RE
										r	r	r	r	r	r

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

Bit 5 **BUSY:** Busy status

This bit defines the status of the SDRAM controller after a Command Mode request

0: SDRAM Controller is ready to accept a new request

1; SDRAM Controller is not ready to accept a new request

Bits 4:3 **MODES2[1:0]:** Status Mode for Bank 2

This bit defines the Status Mode of SDRAM Bank 2.

00: Normal Mode

01: Self-refresh mode

10: Power-down mode

Bits 2:1 **MODES1[1:0]:** Status Mode for Bank 1

This bit defines the Status Mode of SDRAM Bank 1.

00: Normal Mode

01: Self-refresh mode

10: Power-down mode

Bit 0 **RE:** Refresh error flag

0: No refresh error has been detected

1: A refresh error has been detected

An interrupt is generated if REIE = 1 and RE = 1

22.8.6 FMC register map

Table 214. FMC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMC_BCR1	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		WFDIS	CCLKEN	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value	0								0	0	0	0	0	0	0	0	0	0	1	1	0		0	0		1	0	1	1	0	1	1
0x08	FMC_BCR2	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value	0								0	0			0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0

Table 214. FMC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10	FMC_BCR3	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value	0								0	0			0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x18	FMC_BCR4	FMCEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBL SET [1:0]		Res.	Res.	CBURSTRW	CPSIZE [2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	FACCEN	MWID [1:0]		MTYP [1:0]		MUXEN	MBKEN
	Reset value	0								0	0			0	0	0	0	0	0	1	1	0		0	0		1	0	1	0	0	1	0
0x04	FMC_BTR1	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	FMC_BTR2	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	FMC_BTR3	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	FMC_BTR4	DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x20	FMC_ PCSCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTB4EN	CNTB3EN	CNTB2EN	CNTB1EN	CSCOUNT[15:0]															
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	FMC_BWTR1	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x10C	FMC_BWTR2	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x114	FMC_BWTR3	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 214. FMC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x11C	FMC_BWTR4	DATAHLD[1:0]		ACCMOD[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN [3:0]				DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x80	FMC_PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCPS [2:0]				TAR[3:0]				TCLR[3:0]				Res.	Res.	ECCEN	PWID [1:0]		PTYP	PBKEN	PWAITEN	Res.
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0			0	0	1	1	0	0	
0x84	FMC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEMPT	IFEN	ILEN	IREN	IFS	ILS	IRS
	Reset value																										1	0	0	0	0	0	0
0x88	FMC_PMEM	MEMHIZx[7:0]								MEMHOLDx[7:0]								MEMWAITx[7:0]								MEMSETx[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x8C	FMC_PATT	ATTHIZ[7:0]								ATTHOLD[7:0]								ATTWAIT[7:0]								ATTSET[7:0]							
	Reset value	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0x94	FMC_ECCR	ECCx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	FMC_SDCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIPE [1:0]		RBURST	SDCLK [1:0]		WP	CAS [1:0]		NB	MWID [1:0]		NR [1:0]		NC	
	Reset value																		0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
0x144	FMC_SDCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBURST	SDCLK [1:0]		WP	CAS [1:0]		NB	MWID [1:0]		NR [1:0]		NC	
	Reset value																				0	1	1	0	1	0	0	1	0	0	0	0	0
0x148	FMC_SDTR1	Res.	Res.	Res.	Res.	TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]				TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14C	FMC_SDTR2	Res.	Res.	Res.	Res.	TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]				TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x150	FMC_SDCMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD[12:0]														NRFS[3:0]			CTB1	CTB2	MODE[2:0]		
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	FMC_SDRTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REIE	COUNT[12:0]													CRE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158	FMC_SDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	MODES2[1:0]		MODES1[1:0]		Res.	
	Reset value																											0	0	0	0	0	

Refer to *Section 2.3 on page 115* for the register boundary addresses.

23 Octo-SPI interface (OCTOSPI)

23.1 OCTOSPI introduction

The OCTOSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the OCTOSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The OCTOSPI supports the following protocols with associated frame formats:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

23.2 OCTOSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- External (P)SRAM memory support
- Support for single, dual, quad, and octal communication
- Dual memory configuration, where eight bits can be sent/received simultaneously by accessing two quad memories in parallel
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- Asynchronous bus clock versus kernel clock support
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AHB interface with transaction acceptance limited to one: the interface accepts the next transfer on AHB bus only once the previous is completed on memory side.

23.3 OCTOSPI implementation

Table 215. OCTOSPI implementation

OCTOSPI feature	OCTOSPI1
HyperBus standard compliant	X
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AHB compliant data interface	X
Dual AHB interface	X
Asynchronous AHB clock versus kernel clock	X
Functional modes: indirect, automatic status-polling, and memory-mapped	X
Read and write support in memory-mapped mode	X
Dual-quad configuration	X
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS,DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8-, 16-, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
GPDMA interface	X
Dual chip select	
Extended external memory	
Prefetch disable	
Prefetch hardware software	-

23.4 OCTOSPI functional description

23.4.1 OCTOSPI block diagram

Figure 139. OCTOSPI block diagram in octal configuration

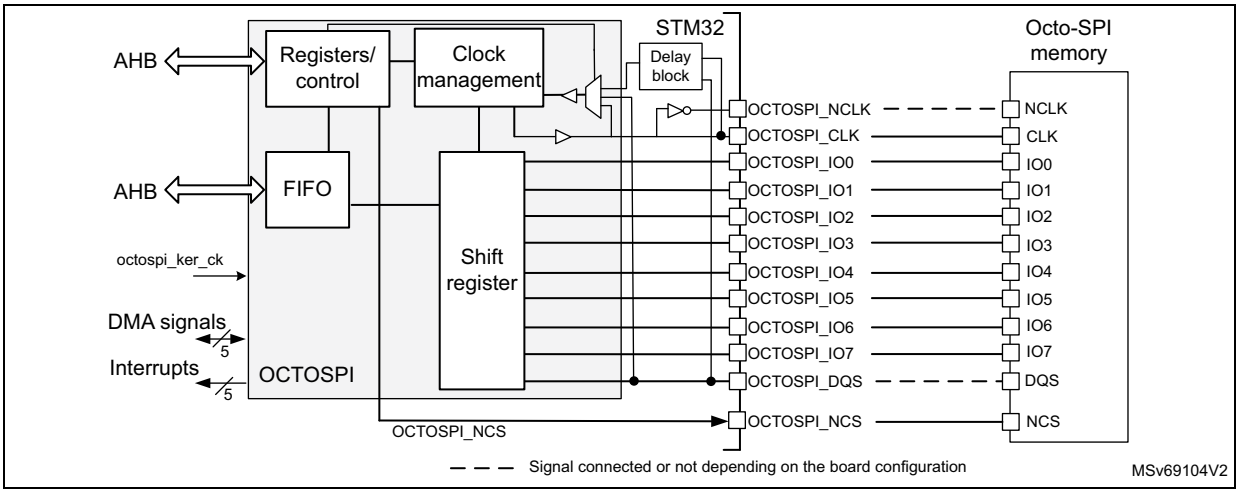


Figure 140. OCTOSPI block diagram in quad configuration

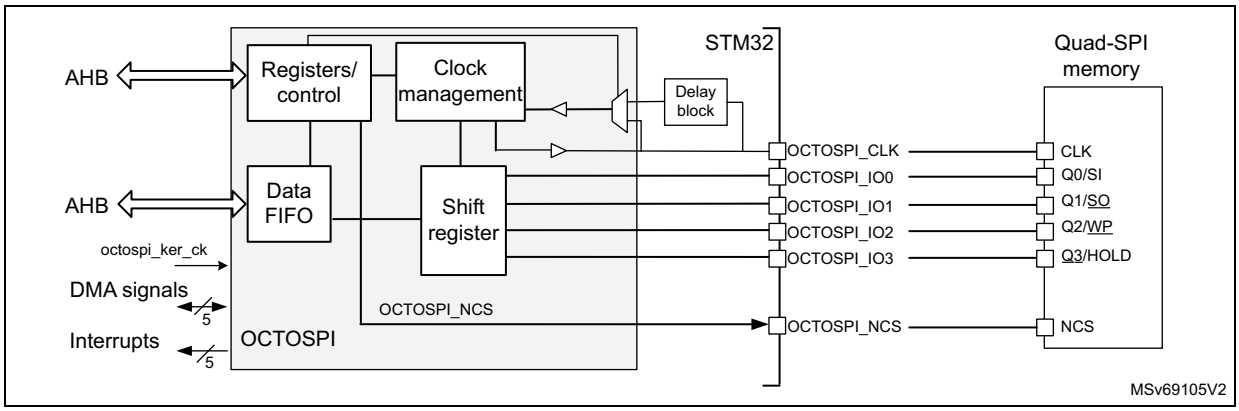
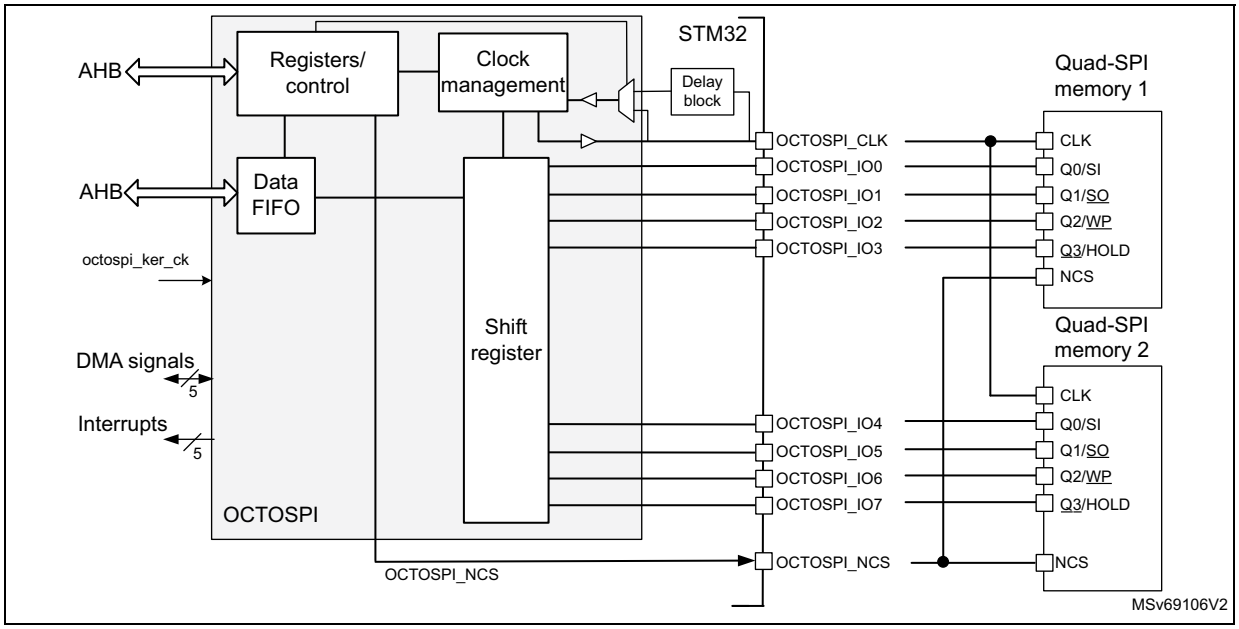


Figure 141. OCTOSPI block diagram in dual-quad configuration



23.4.2 OCTOSPI pins and internal signals

Table 216. OCTOSPI input/output pins

Pin name	Type	Description
OCTOSPI_NCLK	Output	OCTOSPI inverted clock to support 1.8 V HyperBus protocol
OCTOSPI_CLK		OCTOSPI clock
OCTOSPI_IOn (n = 0 to 7)	Input/output	OCTOSPI data pins
OCTOSPI_NCS	Output	Chip select for the memory
OCTOSPI_DQS	Input/output	Data strobe/write mask signal from/to the memory

Caution: Use the same configuration (output speed, HSLV[(a)]) for all OCTOSPI input/output pins to avoid any data corruption.

Table 217. OCTOSPI internal signals

Signal name	Type	Description
octospi_hclk	Input	OCTOSPI AHB clock
octospi_ker_ck	Input	OCTOSPI kernel clock
octospi_dma	N/A	DMA request signal
octospi_it	Output	Global interrupt line (see *Table 221* for the multiple sources of interrupt)

a. When applicable.

23.4.3 OCTOSPI interface to memory modes

The OCTOSPI supports the following protocols:

- regular-command protocol
- HyperBus protocol

The OCTOSPI uses from 6 to 12 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS: data strobe used only in regular-command protocol as input only
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus can be used as possible remap for quad-SPI mode

23.4.4 OCTOSPI regular-command protocol

When in regular-command protocol, the OCTOSPI communicates with the external device using commands. Each command can include the following phases:

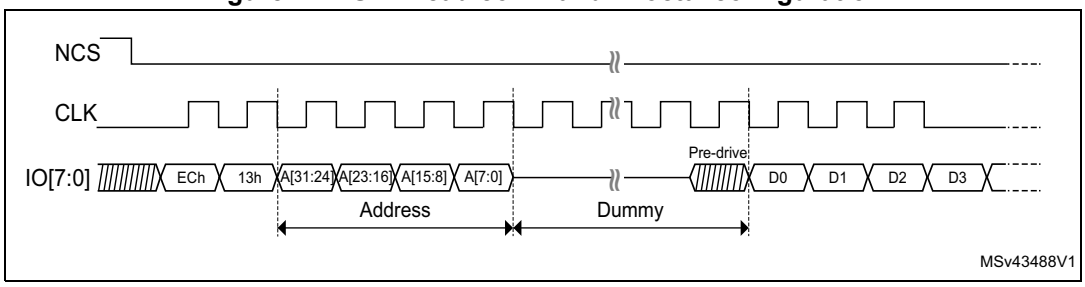
- Instruction phase
- Address phase
- Alternate-byte phase
- Dummy-cycle phase
- Data phase

Any of these phases can be configured to be skipped but, in case of single-phase command, the only use case supported is instruction-phase-only.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operations are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 142. SDR read command in octal configuration



The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured by ISIZE[1:0] in OCTOSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of OCTOSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using IMODE[2:0] of OCTOSPI_CCR.

The instruction can be sent in DTR mode on each rising and falling edge of the clock, by setting IDTR in OCTOSPI_CCR.

When IMODE[2:0] = 000 in OCTOSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

In memory-mapped mode, the instruction used for the write operation is specified in OCTOSPI_WIR, and the instruction format is specified in OCTOSPI_WCCR. The instruction used for the read operation and the instruction format are specified in OCTOSPI_IR and OCTOSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured by ADSIZE[1:0] in OCTOSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified by ADDRESS[31:0] in OCTOSPI_AR. In memory-mapped mode, the address is given directly via the AHB (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ADMODE[2:0] in OCTOSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in OCTOSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in OCTOSPI_WCCR. The address format for the read operation is specified in OCTOSPI_CCR.

Alternate-byte phase

In the alternate-byte phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured by ABSIZE[1:0] in OCTOSPI_CCR. The bytes to be sent are specified in OCTOSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO single in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using ABMODE[2:0] in OCTOSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR in OCTOSPI_CCR.

When ABMODE[2:0] = 000, the alternate-byte phase is skipped and the command sequence proceeds directly to the next phase, if any.

There may be times when only a single nibble needs to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011) and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in OCTOSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0] while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in OCTOSPI_WABR, and the alternate byte format is specified in OCTOSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in OCTOSPI_ABR and OCTOSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified by DCYC[4:0] in OCTOSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-SPI, the quad-SPI, or the octal-SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in OCTOSPI_WTCR. The dummy cycles for the read operation are specified in OCTOSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in OCTOSPI_DLR. In this mode, the data to be sent to the external device must be written to OCTOSPI_DR, while in indirect-read mode the data received from the external device is obtained by reading OCTOSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in OCTOSPI_DLR, and the data received from the external device can be obtained by reading OCTOSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AHB to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)

This can be configured using DMODE[2:0] in OCTOSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR in OCTOSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

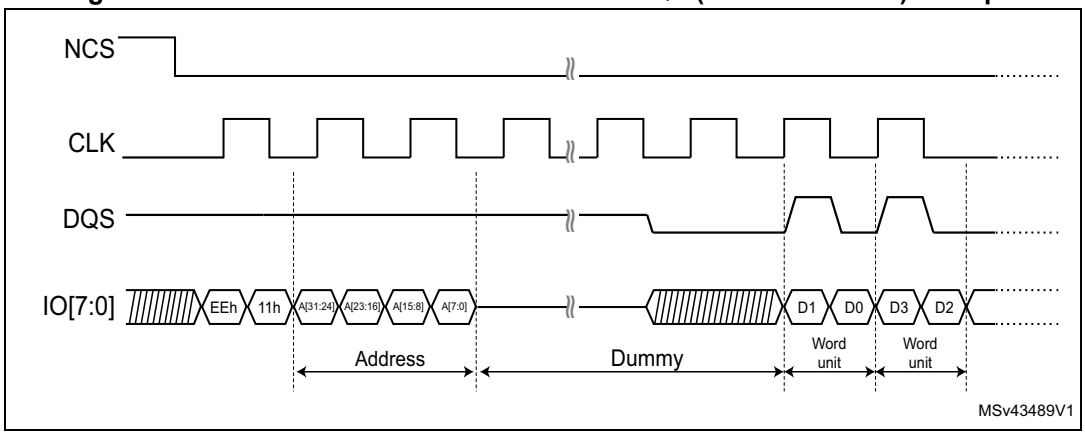
In memory-mapped mode, the data format for the write operation is specified in OCTOSPI_WCCR. The data format for the read operation is specified in OCTOSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE in OCTOSPI_CCR.

Figure 143. DTR read in octal-SPI mode with DQS (Macronix mode) example



23.4.5 OCTOSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the OCTOSPI generating the clock.

The different phases can each be configured separately to use this single-SPI mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0 (to deactivate the “write protect” function).
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO7 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 010, IO0/IO1 are always high-impedance.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO7 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 011, IO0 to IO3 are all high-impedance.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in OCTOSPI_CCR and OCTOSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE[2:0] = 100, IO[0:7] are all high-impedance.

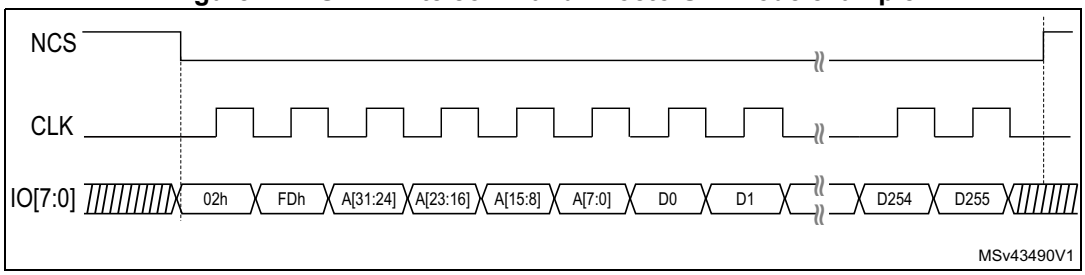
Single-data rate (SDR) mode

By default, all the phases operate in SDR mode.

In this mode, when the OCTOSPI drives the IO0/SO, IO1 to IO7 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the OCTOSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in OCTOSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 144. SDR write command in octo-SPI mode example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in OCTOSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in OCTOSPI_WCCR. The DTR mode for each phase of the read operations is specified in OCTOSPI_CCR.

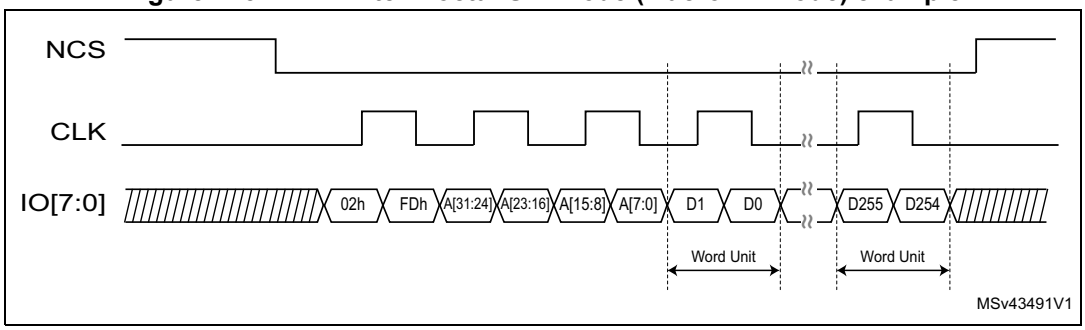
In DTR mode, when the OCTOSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the OCTOSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in OCTOSPI_CCR, the software must clear SSHIFT in OCTOSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

In DTR mode, it is recommended to set DHQC of OCTOSPI_TCR, to shift the outputs by a quarter of cycle and avoid holding issues on the memory side.

Note: DHQC must not be set when the prescaler value is 0, as this action leads to unpredictable behavior.

Figure 145. DTR write in octal-SPI mode (Macronix mode) example



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

Dual-quad configuration

When DMM = 1 in OCTOSPI_CR, the OCTOSPI is in dual-memory configuration: if DMODE = 011, two external Quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either the SDR or DTR mode.

The device size, as specified by DEVSIZE[4:0] in OCTOSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the OCTOSPI gives for address X is the byte at the address X/2 of device A, and the byte that the OCTOSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the OCTOSPI must be configured with a data length of 2 bytes (16 bits), and the OCTOSPI receives one byte from each device.

If each device gives a status of 16 bits, then the OCTOSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The fourth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in OCTOSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

23.4.6 HyperBus protocol

The OCTOSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins depending on the operating voltage:

- IO[7:0] as bidirectional data bus
- DQS for read and write data strobe and latency insertion
- NCS
- CLK
- NCLK for 1.8 V operations (to support this mode, the device must be powered with 1.8 V)

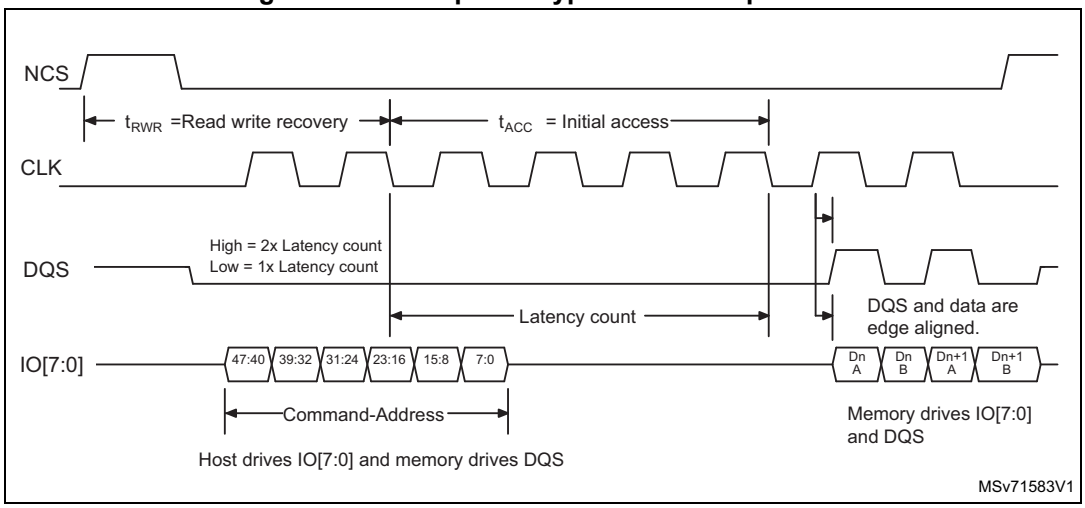
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- Command/address phase
- Data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 146. Example of HyperBus read operation



Note: Due to internal synchronization, up to six extra dummy clock cycles may be generated by the Octo-SPI interface after the last data is read.

The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the OCTOSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 218. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write.
46	Address space	Indicates if the transaction accesses the memory or the register space.
45	Burst type	Indicates if the burst is linear or wrapped.
44-16	Row and upper column address	Selects the row and the upper column addresses.
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page.

The address space is configured through the memory type MTYP[2:0] in OCTOSPI_DCR1.

The total size of the device must be configured through DEVSIZE[4:0] of OCTOSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined by TRWR[7:0] in OCTOSPI_HLCR)
- t_{ACC}: access time for the device (defined by TACC[7:0] in OCTOSPI_HLCR) according to the memory latency

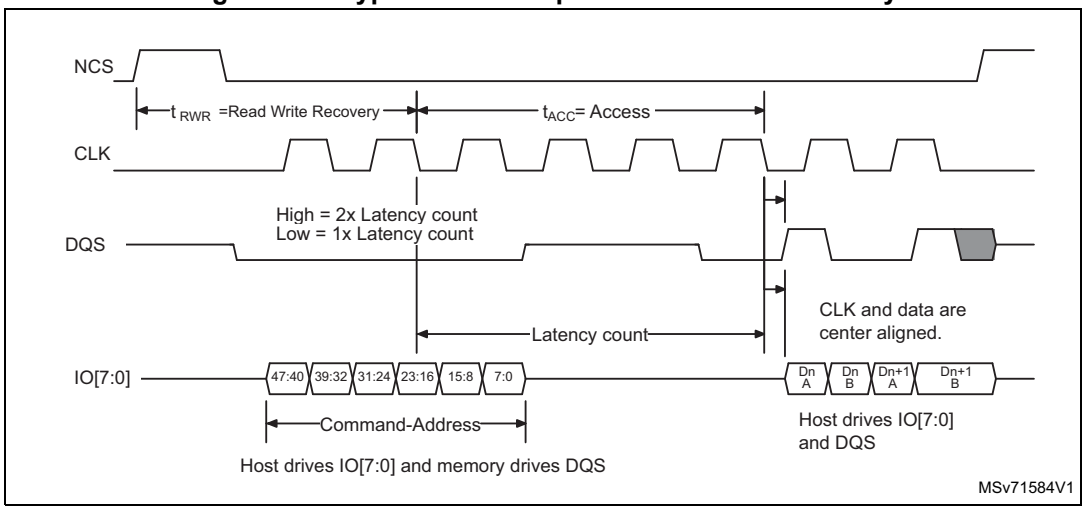
During the read operation, the DQS is used by the device, in two ways (see *Figure 146*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the DQS is used:

- by the device, during the command/address phase, to request an additional latency.
- by the OCTOSPI, during the data phase, for write data masking.

Figure 147. HyperBus write operation with initial latency



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS must be tied to one during one of the DQS signals, during the command/address phase.

An additional t_{ACC} duration is added by the OCTOSPI to meet the device request.

Figure 148. HyperBus read operation with additional latency

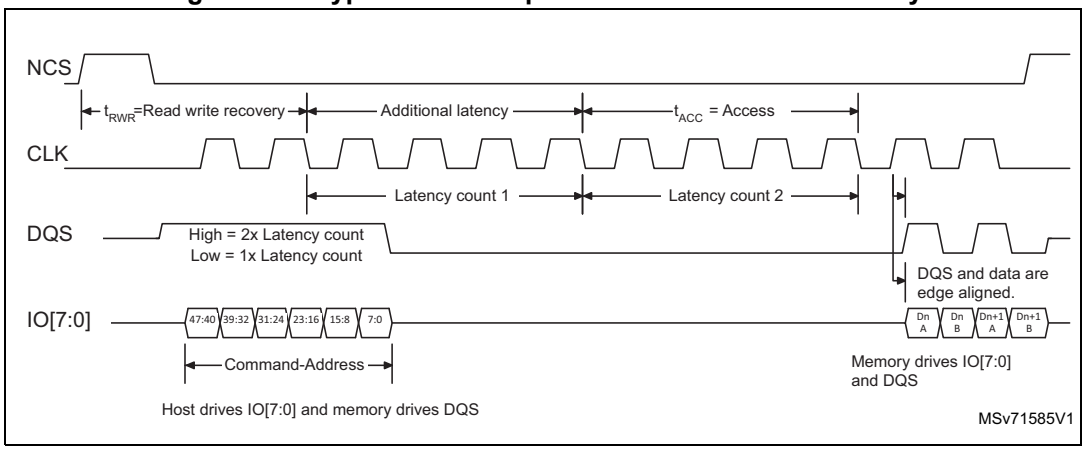
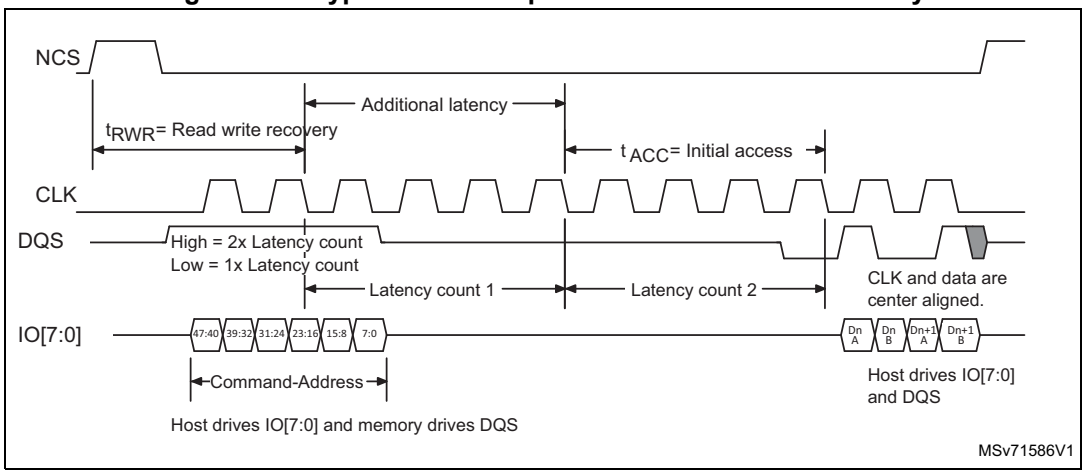


Figure 149. HyperBus write operation with additional latency



Fixed-latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

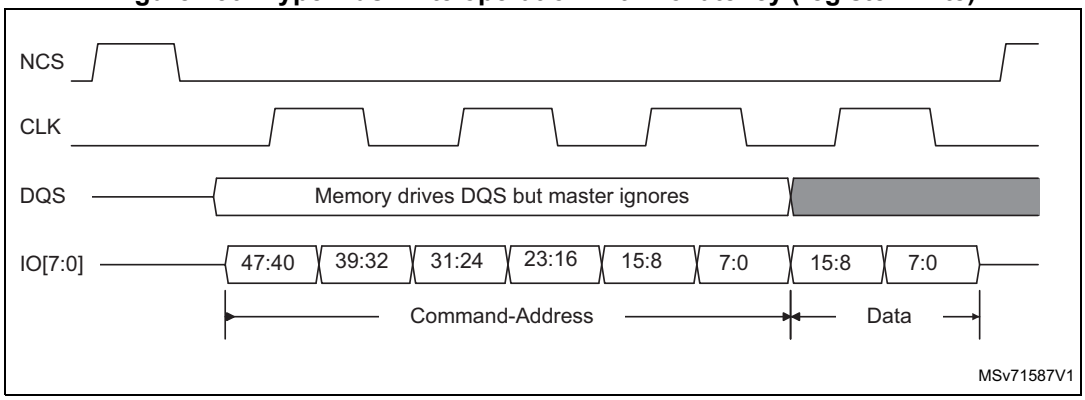
The latency can be forced to 2 x t_{ACC} by setting LM in OCTOSPI_HLCR.

In this OCTOSPI latency mode, the state of the DQS signal is not taken into account by the OCTOSPI, and an additional latency is always added, leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in OCTOSPI_HLCR.

Figure 150. HyperBus write operation with no latency (register write)

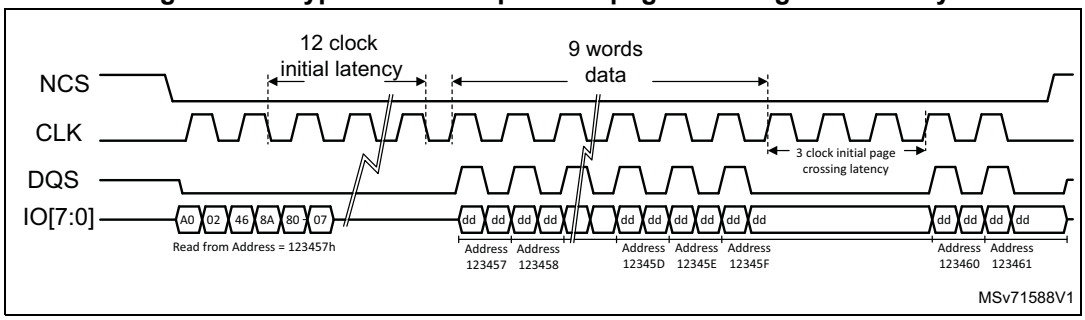


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 151. HyperBus read operation page crossing with latency



23.4.7 Specific features

The OCTOSPI supports some specific features, such as:

- Wrap support
- NCS boundary and refresh

Wrap support

The OCTOSPI supports a hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in OCTOSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 4 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated OCTOSPI_WPxxx registers must be programmed according to the wrapped-read access characteristics. These registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (OCTOSPI_CCR, OCTOSPI_TCR, OCTOSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the OCTOSPI transactions:

- NCS boundary
- Refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with, is configured by CSBOUND[4:0] in OCTOSPI_DCR3: it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. The NCS is then released each time the LSB address is equal to 0xF, and each time a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed by REFRESH[31:0] in OCTOSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command/address/alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

Restarting after an interrupted transfer

When a read or write operation is interrupted by a timeout or communication regulation feature, the Octo-SPI interface, as soon as possible after getting back the port ownership, reissues the initial command sequence together with the address following the last address actually accessed before interruption. The transfer initially set goes on and ends seamlessly.

23.4.8 OCTOSPI operating mode introduction

The OCTOSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

23.4.9 OCTOSPI indirect mode

In indirect mode, the commands are started by writing to the OCTOSPI registers, and data are transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in OCTOSPI_CR, the OCTOSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to OCTOSPI_DR.

When FMODE[1:0] = 01, the OCTOSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading OCTOSPI_DR.

In indirect mode, when the OCTOSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The OCTOSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in OCTOSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the OCTOSPI simply continues to transfer data until it reaches the end of the external device (as

defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in OCTOSPI_CCR.

- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the OCTOSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, the TCF bit is set in OCTOSPI_SR, and an interrupt is generated if TCIE = 1 in OCTOSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in OCTOSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the OCTOSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using the regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data need to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in OCTOSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data need to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in OCTOSPI_AR
- if data need to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in OCTOSPI_DR

A write to OCTOSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in OCTOSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the OCTOSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (OCTOSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01).
- when a write is performed to DATA[31:0] (OCTOSPI_DR) in indirect-write mode (when FMODE = 00).
- when a (dummy) write is performed to INSTRUCTION[31:0] (OCTOSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01).

As soon as a transfer is started, the BUSY bit (OCTOSPI_SR[5]) is automatically set.

FIFO and data management

Data in indirect mode passes through a 32-byte FIFO that is internal to the OCTOSPI. FLEVEL in OCTOSPI_SR indicates how many bytes are currently being held in the FIFO.

AHB burst transactions are supported. Data of the burst are successively written in OCTOSPI_DR, and immediately transferred in the internal FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in OCTOSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to OCTOSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

23.4.10 OCTOSPI automatic status-polling mode

In automatic status-polling mode, the OCTOSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value. The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in OCTOSPI_SR goes high at this point and stays high even between the periodic accesses.

The content of MASK[31:0] in OCTOSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in OCTOSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in OCTOSPI_CR, the AND-match mode is activated: SMF is set in OCTOSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in OCTOSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in OCTOSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the OCTOSPI is disabled (EN = 0).

OCTOSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in OCTOSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. The memory must then be configured in fixed latency.

23.4.11 OCTOSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256 Mbytes range, then an AHB error is given. The effect of this error depends on the AHB master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the OCTOSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the OCTOSPI never stops its prefetch operation. It either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in OCTOSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in OCTOSPI_LPTR, when x cycles have elapsed without access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes. The indirect-write mode fulfills this operation.

However, if the application requires the use of the MCE for encryption (check MCE product availability), the memory-mapped write mode may be used to program encrypted data to external flash memory under the following conditions:

- Prefetch must be enabled.
- In block cipher mode, the CPU must write a complete 128-bit data block to prevent the MCE from initiating read-modify-write operations when only a few bytes need to be

programmed. This precaution avoids incorrect programming operations. There are no specific constraints to respect if the MCE is used in stream cipher mode.

- Apply the abort sequence to exit memory-mapped mode when the data linked to the page has been written in the external memory buffers. The abort sequence triggers the start of the page programming.
- Switch to the automatic status-polling mode to monitor the completion of the page programing phase.
- Relaunch the write enable command in indirect mode, then switch back to the memory-mapped mode configuration to continue to program additional pages if any.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

23.4.12 OCTOSPI configuration introduction

The OCTOSPI configuration is done in three steps:

1. OCTOSPI system configuration
2. OCTOSPI device configuration
3. OCTOSPI mode configuration

23.4.13 OCTOSPI system configuration

The OCTOSPI is configured using OCTOSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- the DMA use with DMAEN
- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in OCTOSPI_LPTR.

The DMA channel must not be enabled during the OCTOSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and OCTOSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

23.4.14 OCTOSPI device configuration

The parameters related to the external device targeted are configured through OCTOSPI_DCR1 and OCTOSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the clock mode with FRCK and CKMODE
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{\text{DEVSIZE+1}}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the OCTOSPI executes two commands, one immediately after the other, it raises the chip-select signal (NCS) high between the two commands for only one CLK cycle by default.

If the external device requires more time between commands, the chip-select high time CSHT[5:0] can be used to specify the minimum number of CLK cycles for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the latency mode must be configured in OCTOSPI_HLCR.

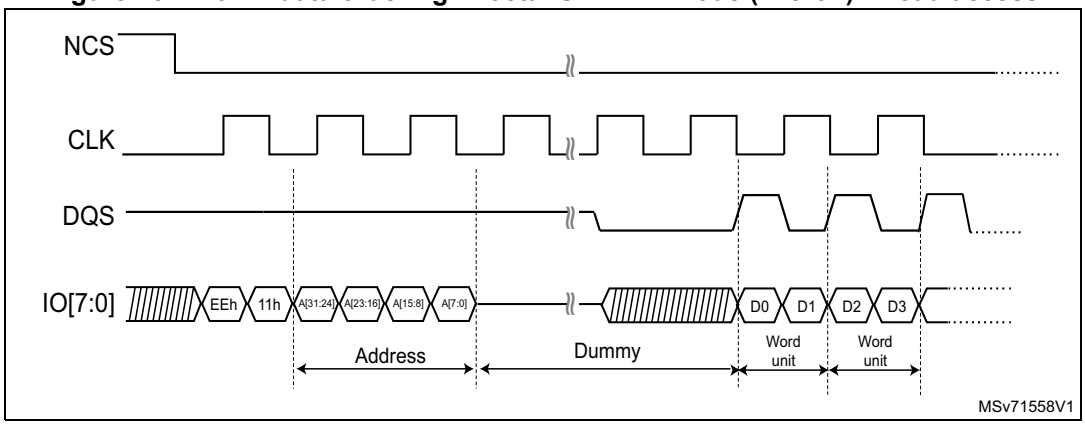
Memory types

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the OCTOSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 152. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the

memory. Refer to *Figure 143: DTR read in octal-SPI mode with DQS (Macronix mode) example*.

- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 219: OctaRAM command address bit assignment (based on 64-Mbyte OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock one when reading data from the memory.

Figure 153. OctaRAM read operation with reverse data ordering D1/D0

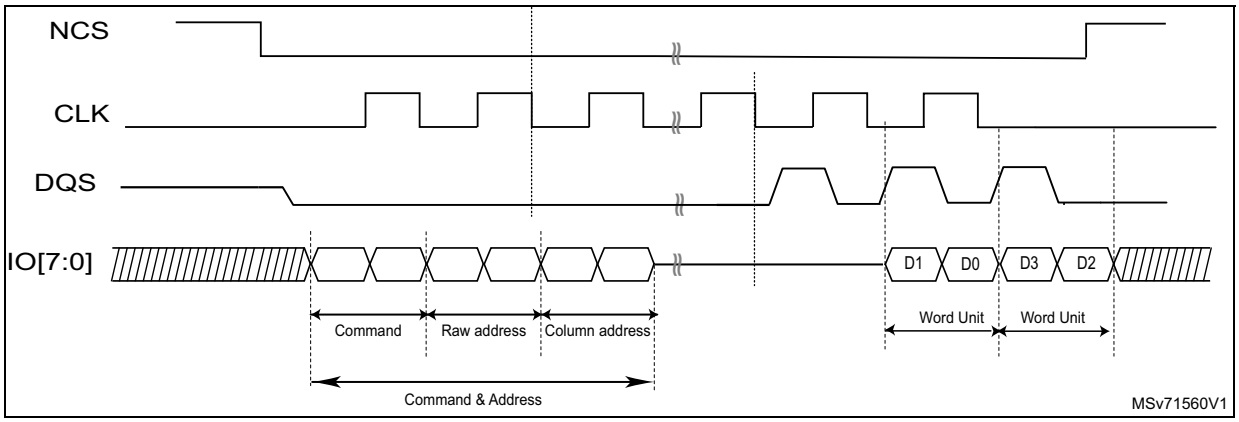


Table 219. OctaRAM command address bit assignment (based on 64-Mbyte(1) OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0(2)

1. Example of 64-Mbyte OctaRAM address assignment:
   Row Address [RA12:RA0]: 8K. Column address [CA9:CA0]: 1K. 64-Mbyte density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes

must be used. The configuration to be set for this particular register space access is MTYP[2:0] = 101.

- Standard mode. It is the mode to use whenever the targeted memory is not corresponding to any others configurations described in this section. MTYP[2:0] = 010 for this standard mode.

23.4.15 OCTOSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected, and data can be read from the external device.

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the frame timing in OCTOSPI_TCR.
3. Specify the frame format in OCTOSPI_CCR.
4. Specify the instruction in OCTOSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
6. Specify the targeted address in OCTOSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

If neither the address register (OCTOSPI_AR) nor the data register (OCTOSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000),
the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and data are retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in OCTOSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in OCTOSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in OCTOSPI_PSMKR, and can be ORed or ANDed with the value stored in OCTOSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The OCTOSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in OCTOSPI_DR.

When the OCTOSPI is used in automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in OCTOSPI_PSMKR.
2. Specify the comparison value in OCTOSPI_PSMAR.
3. Specify the read period in OCTOSPI_PIR.
4. Specify a number of data bytes to read in OCTOSPI_DLR.
5. Specify the frame timing in OCTOSPI_TCR.
6. Specify the frame format in OCTOSPI_CCR.
7. Specify the instruction in OCTOSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR.
9. Specify the optional targeted address in OCTOSPI_AR.

If the address register (OCTOSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as OCTOSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

The memory-mapped mode is entered by setting FMODE[1:0] = 11 in OCTOSPI_CR.

The programmed instruction and frame are sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

When the OCTOSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in OCTOSPI_TCR for read operation.
2. Specify the frame format in OCTOSPI_CCR for read operation.
3. Specify the instruction in OCTOSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_ABR for read operation.
5. Specify the frame timing in OCTOSPI_WTCR for write operation.
6. Specify the frame format in OCTOSPI_WCCR for write operation.

7. Specify the instruction in OCTOSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in OCTOSPI_WABR for write operation.

All configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 has no effect and is not signaled with an error response. On the first access, the OCTOSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

OCTOSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the OCTOSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in OCTOSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

OCTOSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it can be shifted in time to compensate the data propagation delay. This shift is performed by an external delay block located outside the OCTOSPI. The control of this feature depends on the device implementation (see the product reference manual for more details).

In configurations where delay does not need to be compensated, the external delay block can be bypassed by setting DLYBYP in OCTOSPI_DCR1.

23.4.16 OCTOSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the OCTOSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in OCTOSPI_DLR.
2. Specify the targeted address in OCTOSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through OCTOSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to OCTOSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through OCTOSPI_DR.

Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AHB master accesses the memory-mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to OCTOSPI_DR in this mode returns zero.

The data length register (OCTOSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the OCTOSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit, or to abort by setting the ABORT bit.

23.4.17 OCTOSPI error management

The following errors set the TEF flag in OCTOSPI_SR and generates an interrupt if enabled (TEIE = 1 in OCTOSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in OCTOSPI_AR (according to the device size defined by DEVSIZE[4:0]).
- in indirect mode, if the address plus the data length exceed the device size: TEF is set as soon as the access is triggered.

In memory-mapped mode, the OCTOSPI generates an AHB slave error in the following situations,:

- The memory-mapped mode is disabled and an AHB read or write request occurs.
- A read or write address exceeds the size of the external memory.
- An abort is received while a read or write burst is ongoing.
- The OCTOSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in OCTOSPI_WCCR in octal DTR mode, in dual-memory configuration, in HyperBus mode.
- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means that the AHB bus transfer size (HSIZE) is different from 4 bytes (only for memory-mapped mode).
- Bit DMM is set in OCTOSPI_CR (dual-memory configuration) and octal mode is set.

23.4.18 OCTOSPI BUSY and ABORT

Once the OCTOSPI starts an operation with the external device, BUSY is automatically set in OCTOSPI_SR.

In indirect mode, BUSY is reset once the OCTOSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in OCTOSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in OCTOSPI_DR, it is recommended to disable the DMA channel before aborting the OCTOSPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

23.4.19 OCTOSPI reconfiguration or deactivation

Before any OCTOSPI reconfiguration, the software must ensure that all the transactions are completed:

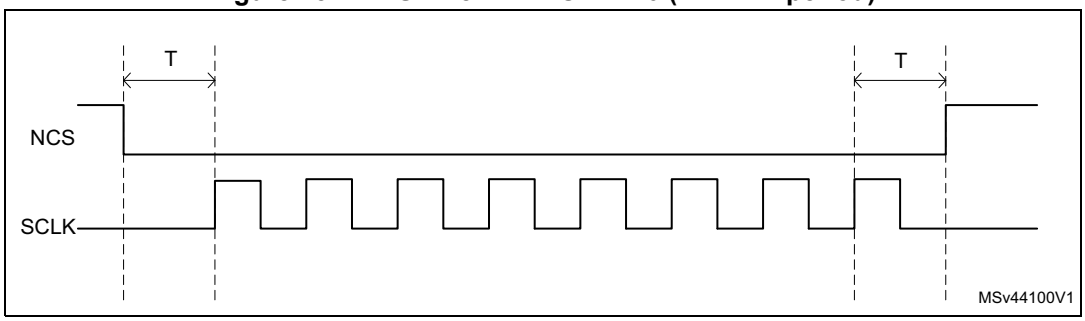
- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier than an abort.

23.4.20 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

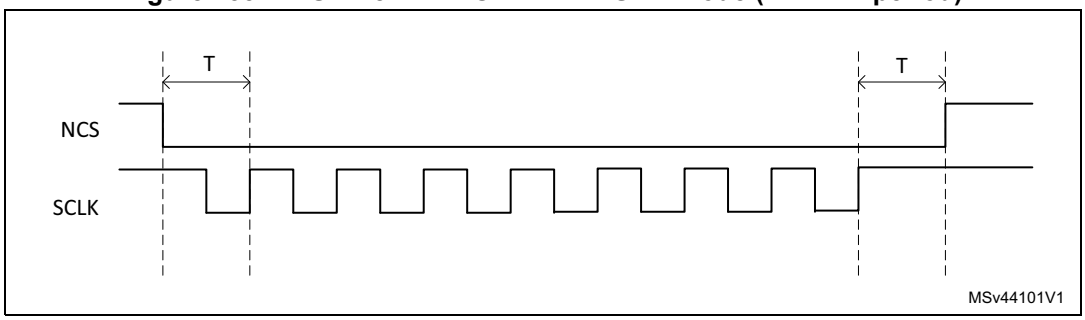
When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 154. NCS when CKMODE = 0 (T = CLK period)



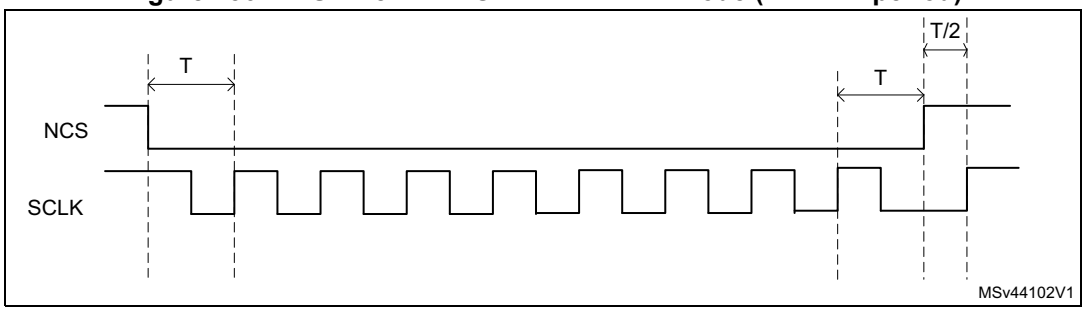
When CKMODE = 1 (clock mode 3: CLK goes high when no operation is in progress) and when in SDR mode, NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 155. NCS when CKMODE = 1 in SDR mode (T = CLK period)



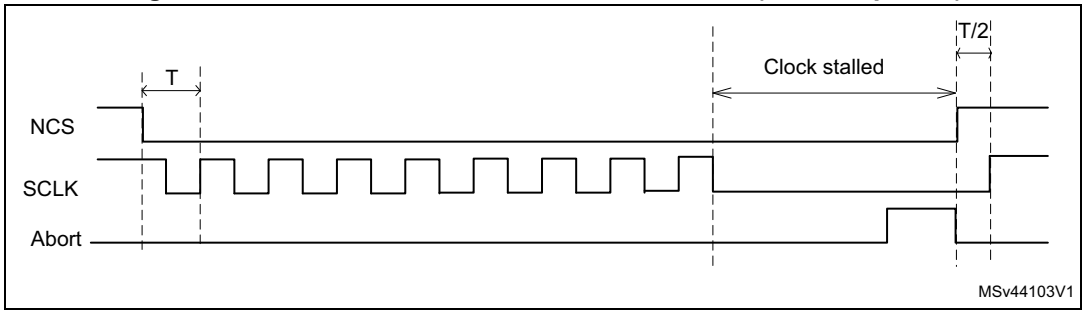
When the CKMODE = 1 (clock mode 3) and DDTR = 1 (data DTR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge (see the figure below). Because the DTR operations must finish with a falling edge, CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

Figure 156. NCS when CKMODE = 1 in DTR mode (T = CLK period)



When the FIFO stays full during a read operation, or if the FIFO stays empty during a write operation, the operation stalls and CLK stays low until the software services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later (see the figure below).

Figure 157. NCS when CKMODE = 1 with an abort (T = CLK period)



23.5 Address alignment and data number

The table below summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 220. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				
Octal memory in DTR mode without WDM(6)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
HyperBus	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.

7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

23.6 OCTOSPI interrupts

An interrupt can be produced on the following events:

- Timeout
- Status match
- FIFO threshold
- Transfer complete
- Transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 221. OCTOSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

23.7 OCTOSPI registers

23.7.1 OCTOSPI control register (OCTOSPI_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FMODE[1:0]		Res.	Res.	Res.	Res.	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
		rw	rw					rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	FTHRES[4:0]					MSEL	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
			rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **FMODE[1:0]**: Functional mode
This bitfield defines the OCTOSPI functional mode of operation.
00: Indirect-write mode
01: Indirect-read mode
10: Automatic status-polling mode (relevant in regular-command protocol only)
11: Memory-mapped mode
If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.
Note: This bitfield can be modified only when BUSY = 0.

Bits 27:24 Reserved, must be kept at reset value.

Bit 23 **PMM**: Polling match mode
This bit indicates which method must be used to determine a match during the automatic status-polling mode.
0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.
Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop
This bit determines if the automatic status-polling mode is stopped after a match.
0: Automatic status-polling mode is stopped only by abort or by disabling the OCTOSPI.
1: Automatic status-polling mode stops as soon as there is a match.
Note: This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable
This bit enables the timeout interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **SMIE**: Status-match interrupt enable
This bit enables the status-match interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **FTHRES[4:0]**: FIFO threshold level

This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in OCTOSPI_SR, to be set.

00000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.

00001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.

...

11111: FTF is set if there are 32 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 32 valid bytes can be read from the FIFO in indirect-read mode.

Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[4:0] value.

Bit 7 **MSEL**: External memory select

This bit selects the external memory to be addressed in single-, dual-, quad-SPI mode in single-memory configuration (when DMM = 0).

0: External memory 1 selected (data exchanged over IO[3:0])

1: External memory 2 selected (data exchanged over IO[7:4])

This bit is ignored when DMM = 1 or when octal-SPI mode is selected.

Bit 6 **DMM**: Dual-memory configuration

This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity

0: Dual-memory configuration disabled

1: Dual-memory configuration enabled

Note: This bit can be modified only when BUSY = 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via OCTOSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the OCTOSPI.

0: OCTOSPI disabled

1: OCTOSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation.
In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

23.7.2 OCTOSPI device configuration register 1 (OCTOSPI_DCR1)

Address offset: 0x0008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. regular-command protocol in single-, dual-, quad- and octal-SPI modes.

Note: In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.

010: Standard mode

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).

100: HyperBus memory mode, the protocol follows the HyperBus specification.

101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect-read/write modes must be used.

Others: Reserved

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:
Number of bytes in device = $2^{[DEVSIZE+1]}$.
DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.
In regular-command protocol, if DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

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

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.
0x0: NCS stays high for at least 1 cycle between external device commands.
0x1: NCS stays high for at least 2 cycles between external device commands.
...
0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **DLYBYP**: Delay block bypass

0: The internal sampling clock (called feedback clock) or the DQS data strobe external signal is delayed by the delay block (for more details on this block, refer to the dedicated section of the reference manual as it is not part of the OCTOSPI peripheral).
1: The delay block is bypassed, so the internal sampling clock or the DQS data strobe external signal is not affected by the delay block. The delay is shorter than when the delay block is not bypassed, even with the delay value set to minimum value in delay block.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.
0: CLK is not free running.
1: CLK is free running (always provided).

Note: Free running clock mode is intended for delay calibration only. No memory or other device access is possible when FRCK is set. Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0/mode 3

This bit indicates the level taken by the CLK between commands (when NCS = 1).
0: CLK must stay low while NCS is high (chip-select released). This is referred to as clock mode 0.
1: CLK must stay high while NCS is high (chip-select released). This is referred to as clock mode 3.

23.7.3 OCTOSPI device configuration register 2 (OCTOSPI_DCR2)

Address offset: 0x000C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This bitfield indicates the wrap size to which the memory is configured. For memories which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in the OCTOSPI1_WPIR register.

000: Wrapped reads are not supported by the memory.

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as OCTOSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the OCTOSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50 %. The clock signal remains low one cycle longer than it stays high.

23.7.4 OCTOSPI device configuration register 3 (OCTOSPI_DCR3)

Address offset: 0x0010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This bitfield enables the transaction boundary feature. When active, a minimum value of 3 is recommended. The NCS is released on each boundary of $2^{CSBOUND}$ bytes.

0: NCS boundary disabled

Others: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:0 Reserved, must be kept at reset value.

23.7.5 OCTOSPI device configuration register 4 (OCTOSPI_DCR4)

Address offset: 0x0014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual- or quad-SPI mode, because the byte transmission must be completed.

0: Refresh disabled

Others: Maximum communication length is set to REFRESH + 1 clock cycles.

Note: REFRESH count is based on the divided clock period: if OCTOSPI_DCR2 PRESCALER bitfield is changed, the REFRESH field must be updated accordingly.

23.7.6 OCTOSPI status register (OCTOSPI_SR)

Address offset: 0x0020

Reset value: 0x0000 0000

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **FLEVEL[5:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 32 when it is full.

In automatic status-polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (OCTOSPI_PSMAR).

It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.

It is cleared automatically as soon as the threshold condition is no longer true.

In automatic status-polling mode, this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode.

It is cleared by writing 1 to CTEF.

23.7.7 OCTOSPI flag clear register (OCTOSPI_FCR)

Address offset: 0x0024

Reset value: 0x0000 0000

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

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

Bit 4 **CTOF**: Clear timeout flag
Writing 1 clears the TOF flag in the OCTOSPI_SR register.

Bit 3 **CSMF**: Clear status match flag
Writing 1 clears the SMF flag in the OCTOSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag
Writing 1 clears the TCF flag in the OCTOSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag
Writing 1 clears the TEF flag in the OCTOSPI_SR register.

23.7.8 OCTOSPI data length register (OCTOSPI_DLR)

Address offset: 0x0040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value+1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.

All 1's in indirect mode means undefined length, where OCTOSPI continues until the end of the memory, as defined by DEVSIZE.

0x0000_0000: 1 byte is to be transferred.

0x0000_0001: 2 bytes are to be transferred.

0x0000_0002: 3 bytes are to be transferred.

0x0000_0003: 4 bytes are to be transferred.

...

0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.

0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.

0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.

DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.

This bitfield has no effect in memory-mapped mode.

23.7.9 OCTOSPI address register (OCTOSPI_AR)

Address offset: 0x0048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

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

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In HyperBus protocol, this field must be even as this protocol is 16-bit word oriented. In dual-memory configuration, AR[0] is forced to 0.

23.7.10 OCTOSPI data register (OCTOSPI_DR)

Address offset: 0x0050

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.

Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

23.7.11 OCTOSPI polling status mask register (OCTOSPI_PSMKR)

Address offset: 0x0080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.

23.7.12 OCTOSPI polling status match register (OCTOSPI_PSMAR)

Address offset: 0x0088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **MATCH[31:0]**: Status match
Value to be compared with the masked status register to get a match

23.7.13 OCTOSPI polling interval register (OCTOSPI_PIR)

Address offset: 0x0090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval
Number of CLK cycles between a read during the automatic status-polling phases

23.7.14 OCTOSPI communication configuration register (OCTOSPI_CCR)

Address offset: 0x0100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for data phase
1: DTR mode enabled for data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate- byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

23.7.15 OCTOSPI timing configuration register (OCTOSPI_TCR)

Address offset: 0x0108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
0: No delay hold
1: 1/4 cycle hold

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

23.7.16 OCTOSPI instruction register (OCTOSPI_IR)

Address offset: 0x0110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

23.7.17 OCTOSPI alternate bytes register (OCTOSPI_ABR)

Address offset: 0x0120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address.

23.7.18 OCTOSPI low-power timeout register (OCTOSPI_LPTR)

Address offset: 0x00130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the OCTOSPI prefetches the subsequent bytes and hold them in the FIFO.

This bitfield indicates how many CLK cycles the OCTOSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

23.7.19 OCTOSPI wrap communication configuration register (OCTOSPI_WPCCR)

Address offset: 0x0140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		AD DTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management.

0: DQS disabled

1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: no alternate bytes
001: alternate bytes on a single line
010: alternate bytes on two lines
011: alternate bytes on four lines
100: alternate bytes on eight lines
Others: reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for address phase
1: DTR mode enabled for address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

23.7.20 OCTOSPI wrap timing configuration register (OCTOSPI_WPTCR)

Address offset: 0x0148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the OCTOSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The firmware must assure that SSHIFT=0 when the data phase is configured in DTR mode (when DDTR = 1).

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DHQC**: Delay hold quarter cycle
Add a quarter cycle delay on the outputs in DTR communication to match hold requirement.
0: No quarter cycle delay
1: 1/4 cycle delay inserted

Bits 27:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

23.7.21 OCTOSPI wrap instruction register (OCTOSPI_WPIR)

Address offset: 0x0150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

23.7.22 OCTOSPI wrap alternate bytes register (OCTOSPI_WPABR)

Address offset: 0x0160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

23.7.23 OCTOSPI write communication configuration register (OCTOSPI_WCCR)

Address offset: 0x0180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate bytes double transfer rate
This bit sets the DTR mode for the alternate-bytes phase.
0: DTR mode disabled for alternate-bytes phase
1: DTR mode enabled for alternate-bytes phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for instruction phase
1: DTR mode enabled for instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

23.7.24 OCTOSPI write timing configuration register (OCTOSPI_WTCR)

Address offset: 0x0188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

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

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

23.7.25 OCTOSPI write instruction register (OCTOSPI_WIR)

Address offset: 0x0190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

23.7.26 OCTOSPI write alternate bytes register (OCTOSPI_WABR)

Address offset: 0x01A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes

Optional data to be sent to the external SPI device right after the address

23.7.27 OCTOSPI HyperBus latency configuration register (OCTOSPI_HLCR)

Address offset: 0x0200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TRWR[7:0]:** Read-write minimum recovery time

Device read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7:0]**: Access time

Device access time according to the memory latency, expressed in number of communication clock cycles

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

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.

0: Latency on write accesses

1: No latency on write accesses

Bit 0 **LM:** Latency mode
This bit selects the latency mode.
0: Variable initial latency
1: Fixed latency

23.7.28 OCTOSPI register map

Table 222. OCTOSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	OCTOSPI_CR	Res.	Res.	FMDOE[1:0]		Res.	Res.	Res.	Res.	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	Res.	FTHRES[4:0]					MSEL	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
	Reset value			0	0					0	0		0	0	0	0	0				0	0	0	0	0	0	0			0	0	0	0
0x0004	Reserved	Reserved																															
0x0008	OCTOSPI_DCR1	Res.	Res.	Res.	Res.	Res.	MTYP [2:0]			Res.	Res.	Res.	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	DLYBYP	Res.	FRCK	CKMODE
	Reset value						0	0	0				0	0	0	0	0			0	0	0	0	0	0					0		0	0
0x000C	OCTOSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x0010	OCTOSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0	0	0	0	0																
0x0014	OCTOSPI_DCR4	REFRESH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018-0x001C	Reserved	Reserved																															
0x0020	OCTOSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[5:0]						Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																			0	0	0	0	0	0			0	0	0	0	0	0
0x0024	OCTOSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																												0	0		0	0
0x0028-0x003C	Reserved	Reserved																															
0x0040	OCTOSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	Reserved	Reserved																															
0x0048	OCTOSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															

Table 222. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0050	OCTOSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054-0x007C	Reserved	Reserved																															
0x0080	OCTOSPI_PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0084	Reserved	Reserved																															
0x0088	OCTOSPI_PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008C	Reserved	Reserved																															
0x0090	OCTOSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0094-0x00FC	Reserved	Reserved																															
0x0100	OCTOSPI_CCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0104	Reserved	Reserved																															
0x0108	OCTOSPI_TCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																							0	0	0	0	0
0x010C	Reserved	Reserved																															
0x0110	OCTOSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114-0x011C	Reserved	Reserved																															
0x0120	OCTOSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0124-0x012C	Reserved	Reserved																															
0x0130	OCTOSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134-0x013C	Reserved	Reserved																															
0x0140	OCTOSPI_WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value	0		0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0144	Reserved	Reserved																															

Table 222. OCTOSPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0148	OCTOSPI_ WPTCR	Res.	SSHIFT	Res.	DHQC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0		0																								0	0	0	0	0
0x014C	Reserved	Reserved																															
0x0150	OCTOSPI_WPIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0154- 0x015C	Reserved	Reserved																															
0x0160	OCTOSPI_ WPABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0164- 0x017C	Reserved	Reserved																															
0x0180	OCTOSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value	0		0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x0184	Reserved	Reserved																															
0x0188	OCTOSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x018C	Reserved	Reserved																															
0x0190	OCTOSPI_WIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0194- 0x019C	Reserved	Reserved																															
0x01A0	OCTOSPI_WABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01A4- 0x01FC	Reserved	Reserved																															
0x0200	OCTOSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

24 Secure digital input/output MultiMediaCard interface (SDMMC)

24.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 5.1*.
  Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit.
  (HS200 SDMMC_CK speed limited to maximum allowed I/O speed)(HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 6.0*.
  (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.*
  Card support for two different databus modes: 1-bit (default) and 4-bit.
  (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode.
  (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

24.2 SDMMC implementation

Table 223. SDMMC instances on device(1)

Devices	SDMMC1	SDMMC2
STM32H562/563/573xx	X	X
STM32H523/533	X	-

1. X = instance is available.

Table 224. SDMMC features

SDMMC modes/features(1)	SDMMC1	SDMMC2
Variable delay (SRD104, HS200)	X	X
SDMMC_CKIN	X	X
SDMMC_CDIR, SDMMC_D0DIR	X	-
SDMMC_D123DIR	X	-

1. X = supported.

24.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 158. SDMMC "no response" and "no data" operations

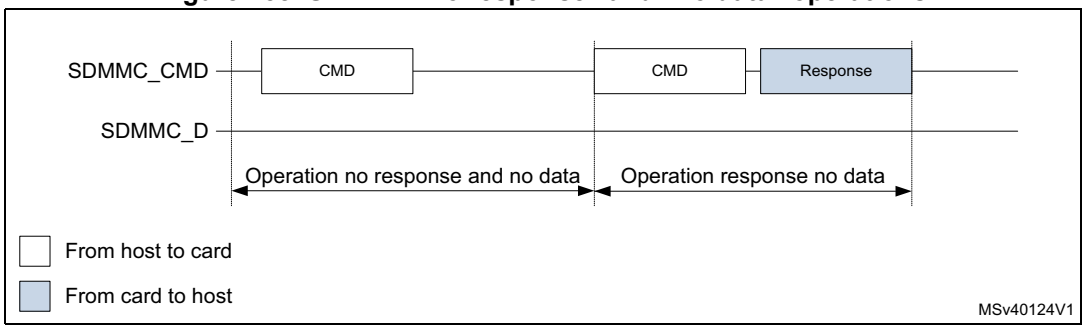
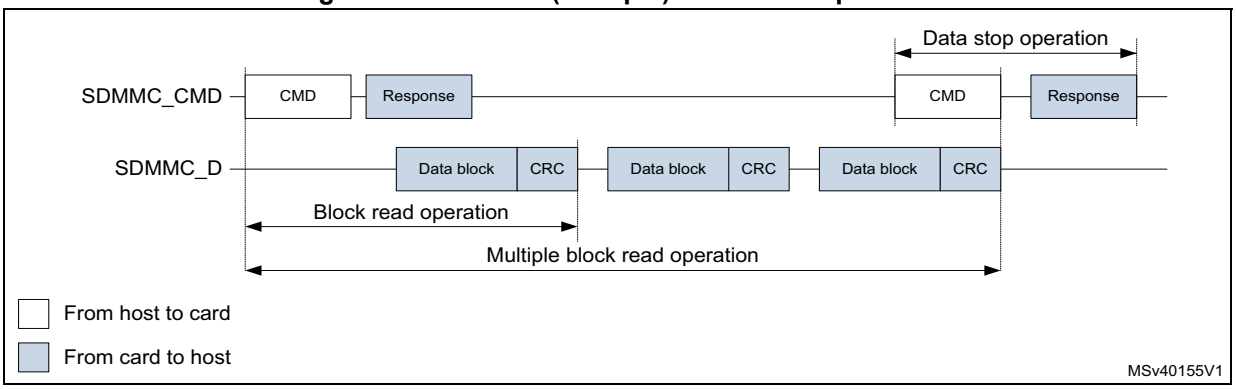
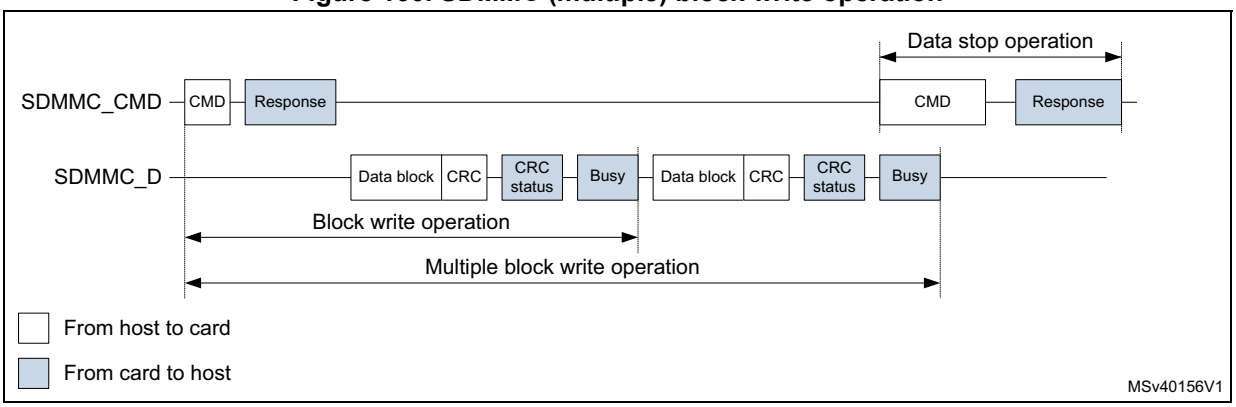


Figure 159. SDMMC (multiple) block read operation



*Note: The Stop Transmission command is not required at the end of a e•*MMC *multiple block read with predefined block count.*

Figure 160. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 161. SDMMC (sequential) stream read operation

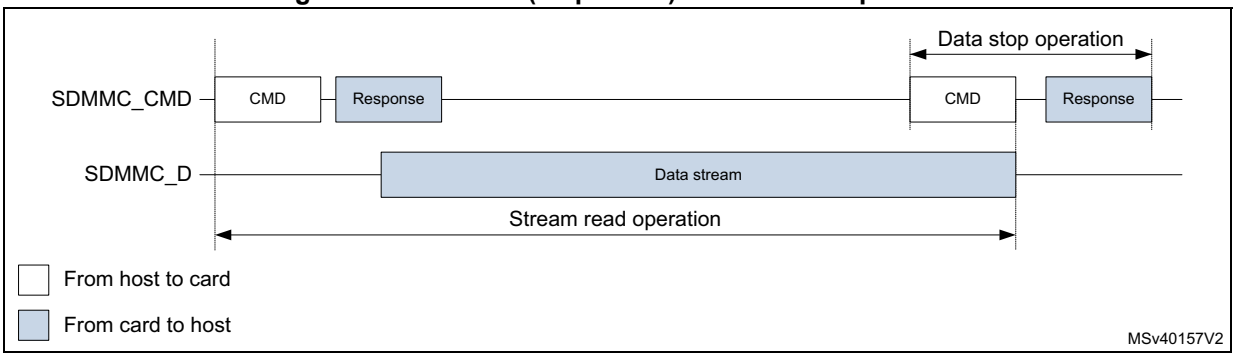
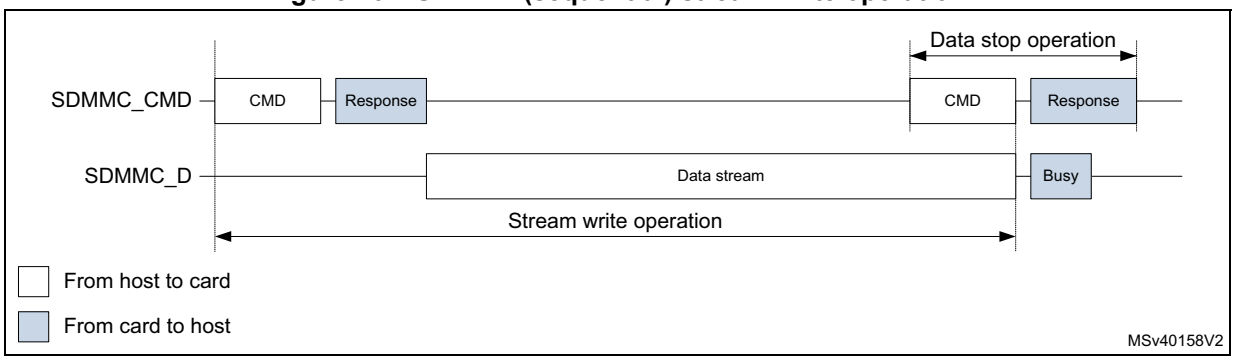


Figure 162. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

24.4 SDMMC operation modes

Table 225. SDMMC operation modes SD and SDIO

SDIO bus speed modes(1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V]
DS (default speed)	12.5	25	3.3
HS (high speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8
SDR104	104	208	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 226. SDMMC operation modes *e*•MMC

e•MMC bus speed modes (1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V](5)
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V
High speed HS200	200	200	1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.

24.5 SDMMC functional description

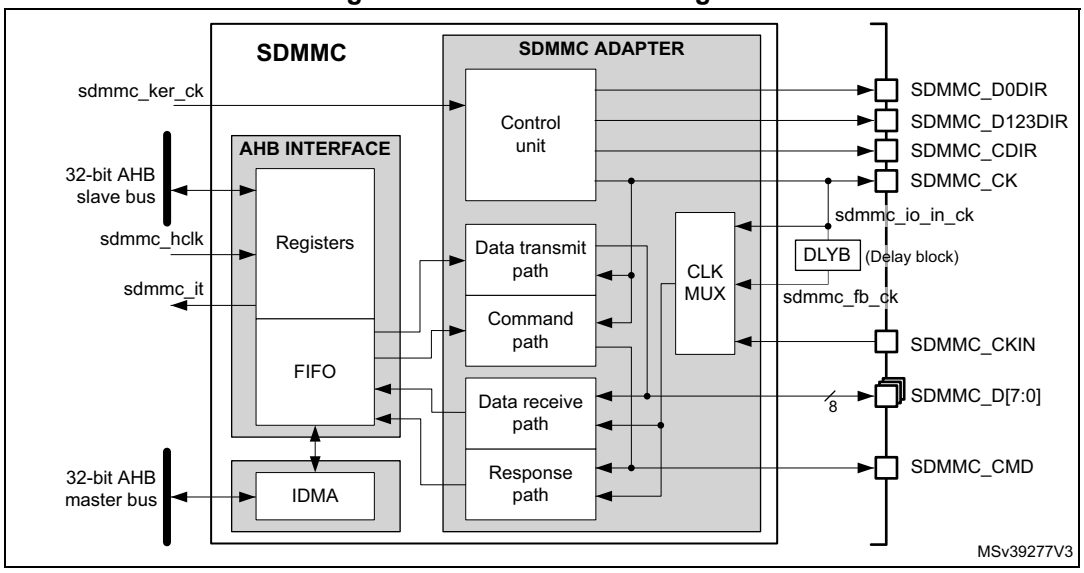
The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the *e*•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

24.5.1 SDMMC block diagram

Figure 163 shows the SDMMC block diagram.

Figure 163. SDMMC block diagram



24.5.2 SDMMC pins and internal signals

Table 227 lists the SDMMC internal input/output signals, *Table 228* the SDMMC pins (alternate functions).

Table 227. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt

Table 227. SDMMC internal input/output signals (continued)

Signal name	Signal type	Description
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 228. SDMMC pins

Pin name	Pin type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

24.5.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 164*.

Figure 164. SDMMC Command and data phase relation

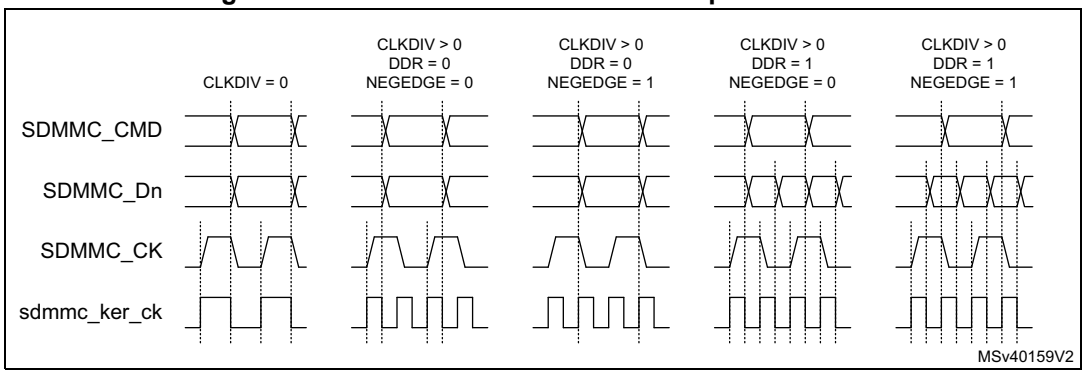


Table 229. SDMMC Command and data phase selection

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	Generated on sdmmc_ker_ck falling edge	
>0	0	0	Generated on sdmmc_ker_ck rising edge	Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	Generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

24.5.4 SDMMC adapter

The SDMMC adapter (see *Figure 163: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

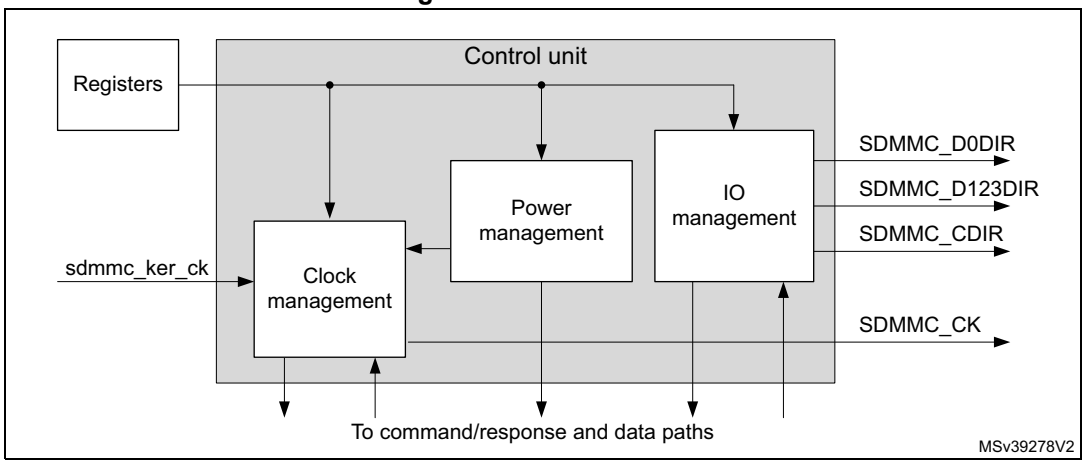
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 165*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 165. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for flow control, for example.

The clock outputs are inactive:

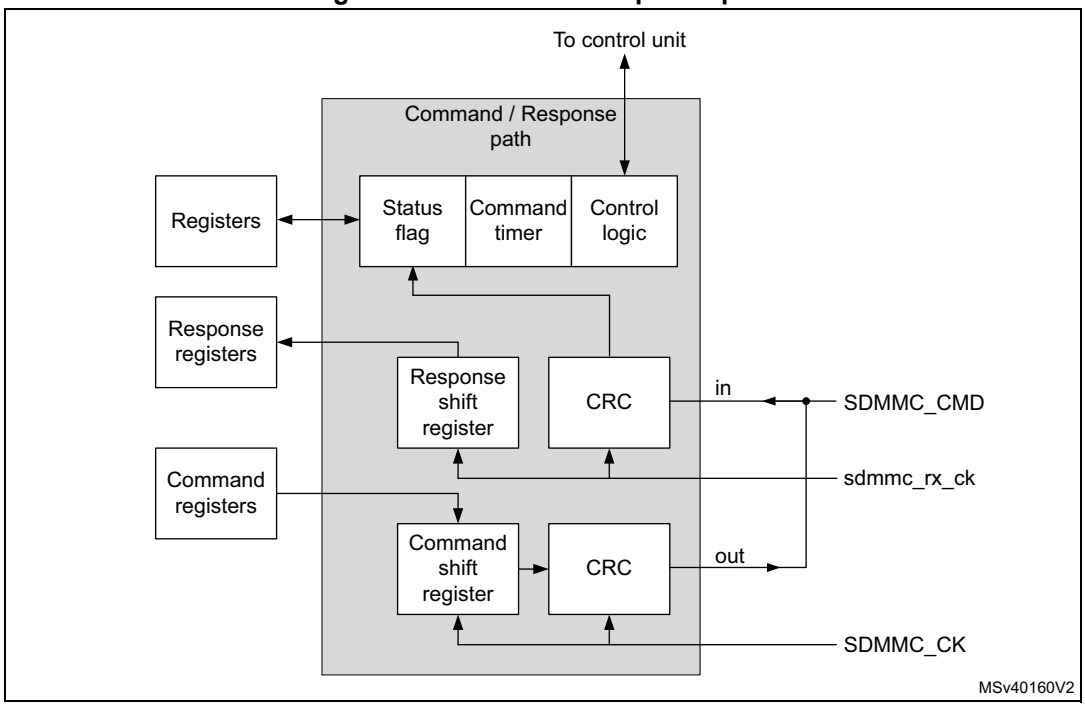
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card,. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 166. Command/response path

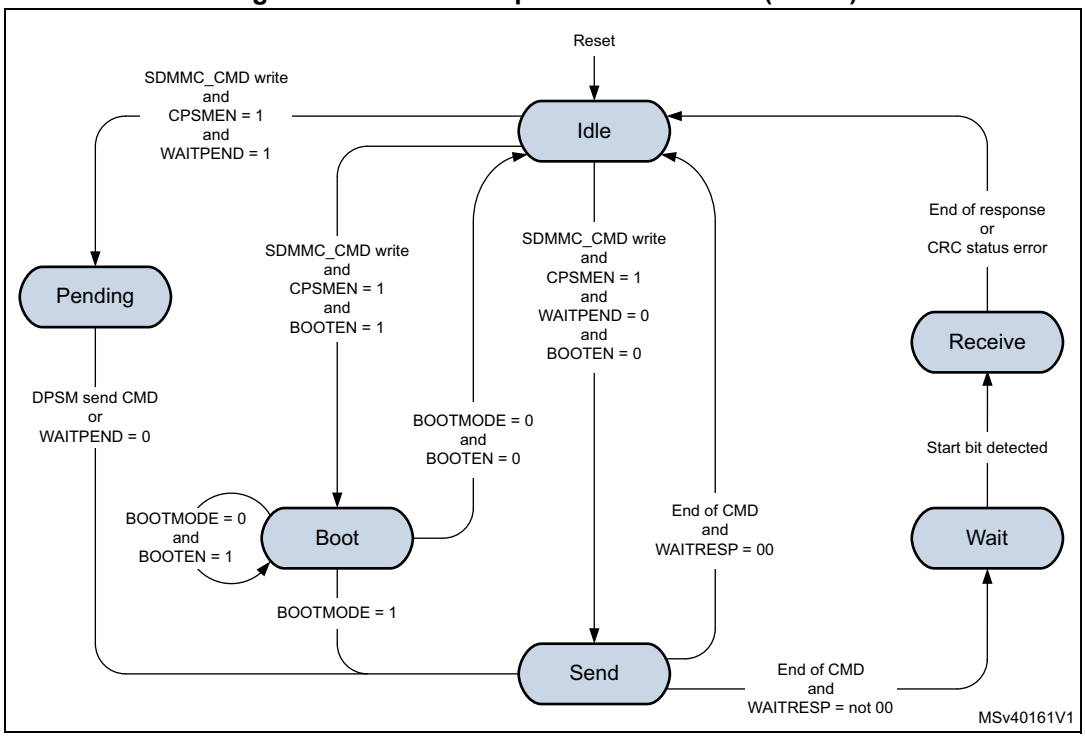


Command/response path state machine (CPSM):

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active (not in the Idle state), the CPSMACT bit is set.

Figure 167. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 and BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the

CMDSENT flag is generated immediately after the command end bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
  a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
  a) When a start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.
- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH ≤ 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the

DPSM before moving to the Send state. This enables, for example, the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 230.*

The command token data is taken from two registers, one containing a 32-bit argument and the other containing the 6-bit command index (six bits sent to a card).

Table 230. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 231*, *Table 232* and *Table 233*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in five registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 231. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 232. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 233. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 164*.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

CRC[6:0] = remainder [(M(x) * x^7) / G(x)]

G(x) = $x^7 + x^3 + 1$

M(x) = (first bit) * x^n + (second bit) * x^{n-1}+... + (last bit before CRC) * x^0

Where n = 39 or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 234*.

Table 234. Specific Commands overview

VSWITCH	BOOTEN	BOOTMOD	CMDTRAN	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start voltage switch sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The command/response path implements the status flags and associated clear bits shown in *Table 235*:

Table 235. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response (CPSM moves from Send to Idle).
CMDREND	Set at the end of the command response when the CRC is OK (CPSM moves from Receive to Idle).
CCRCFAIL	Set at the end of the command response when the CRC is FAIL (CPSM moves from Receive to Idle).
CTIMEOUT	Set after the command when no response start bit received before the timeout (CPSM moves from Wait to Idle).
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped (no impact on CPSM).
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms (no impact on CPSM).
CPSMACT	Command transfer in progress (CPSM not in Idle state).

The command path error handling is shown in *Table 236*:

Table 236. Command path error handling

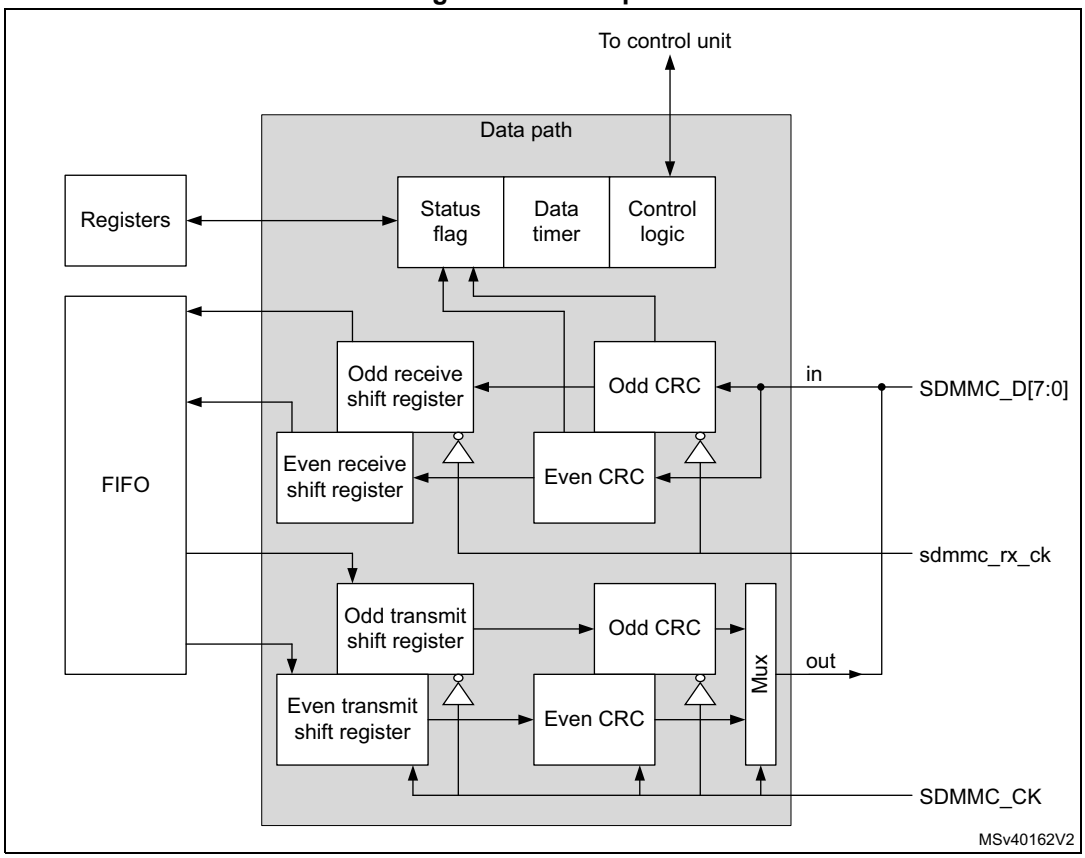
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 168* shows the data path block diagram.

Figure 168. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 169* and *Figure 170*:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode, the SDMMC_CK clock division must be ≥ 2.

Figure 169. DDR mode data packet clocking

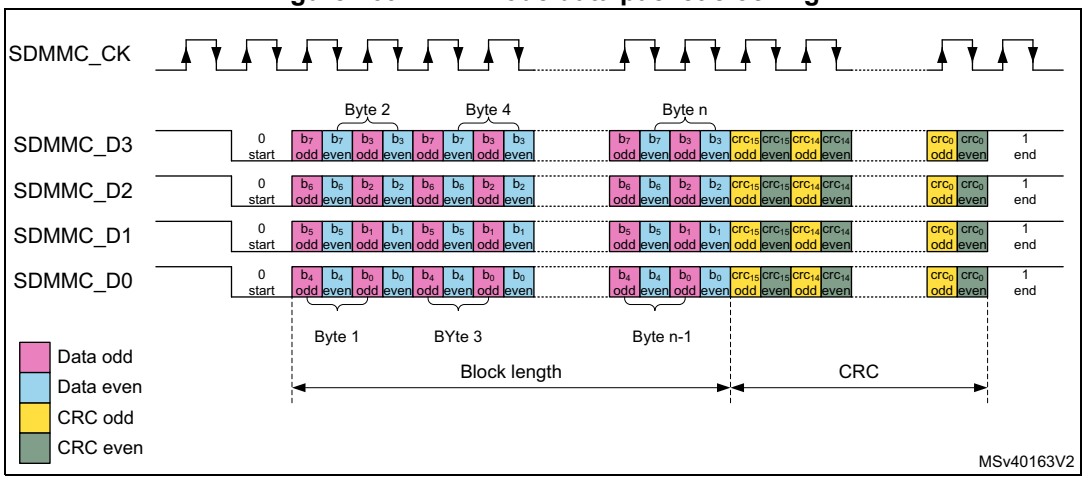
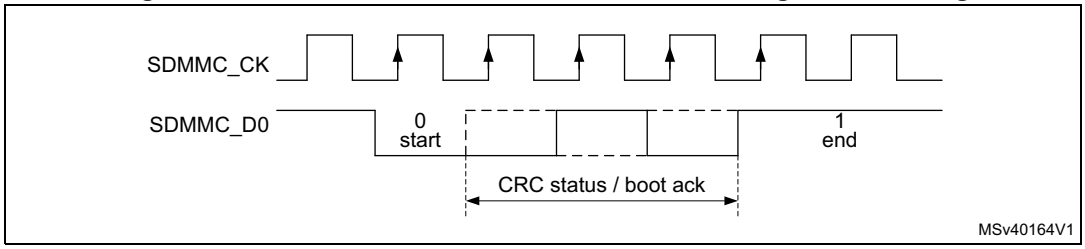


Figure 170. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

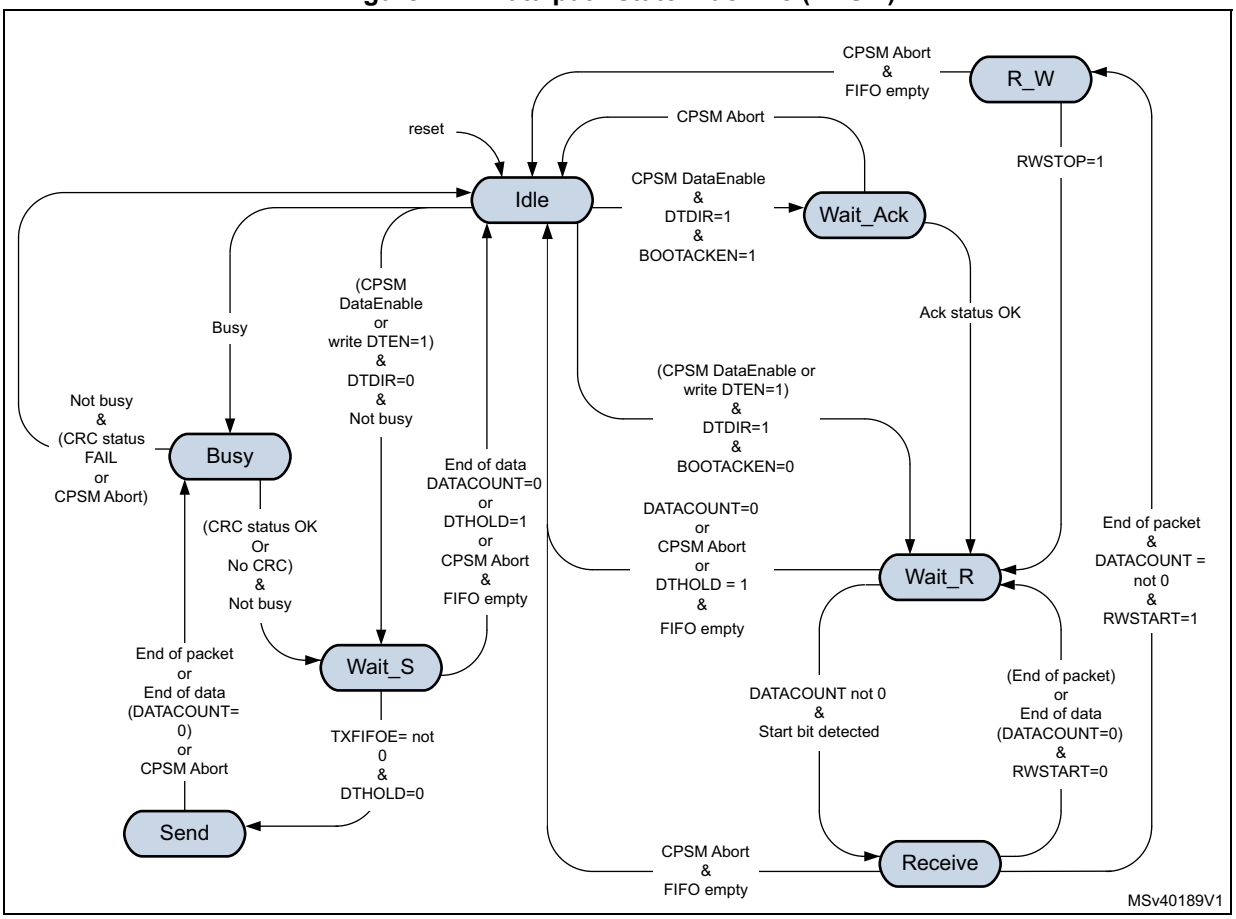
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 171*. When ever the DPSM is active (not in the Idle state), the DPSMACT bit is set.

Figure 171. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

  – When the data direction bit (DTDIR) indicates send, moves to the Wait_S.

  – When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.

Note: DTEN must not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.

- When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.

– When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  – The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  – If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  – In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  – In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.
  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
  - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
  - Wait_R state otherwise.
  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
  - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
  - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  – In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  – When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  – When an CPSM Abort signal is received:
  - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  – When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  - If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  - When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  - When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  - In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command (CMD12 Stop Transmission command).
  - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  - When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT = 0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer

with DATACOUNT = 0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0:

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 237. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	Yes	1	00
SDIO multibyte	0	(DATALENGTH)	Yes	1	01
e•MMC stream	0	(DATALENGTH)	No	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 238*:

Table 238. Data path status flags and clear bits

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)
	Boot	

Table 238. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)
	Boot	
ACKFAIL	Boot	Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)
	Boot	
ACKTIMEOUT	Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DBCKEND	TX	When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from Busy to Wait_S)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from Receive to R_W)
	Boot	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from Wait_R to Idle)
DABORT	CMD R1b	When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)
	TX	When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)
	RX	
	Boot	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 239*:

Table 239. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 240*.

Table 240. Data FIFO access

Data FIFO access	IDMAEN
From firmware via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   – Configures the FIFO in transmit mode.
3. Enable the data transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   – The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.

- The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.

5. Write data to the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 241*.

Table 241. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   – The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
5. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
6. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 242*.

Table 242. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

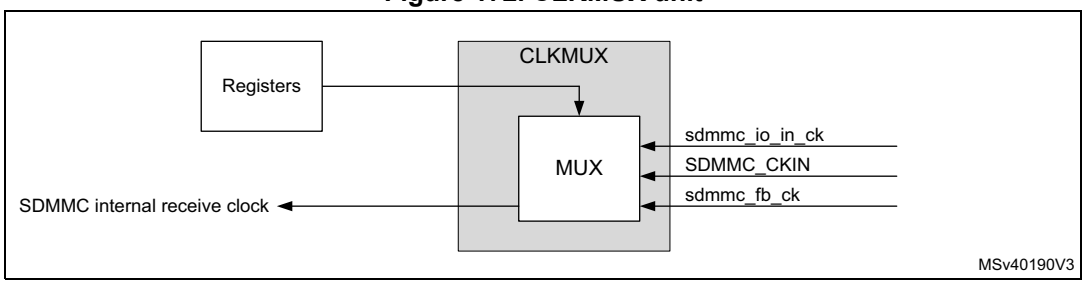
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 172. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

24.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.
- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

24.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer.

When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGHT, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

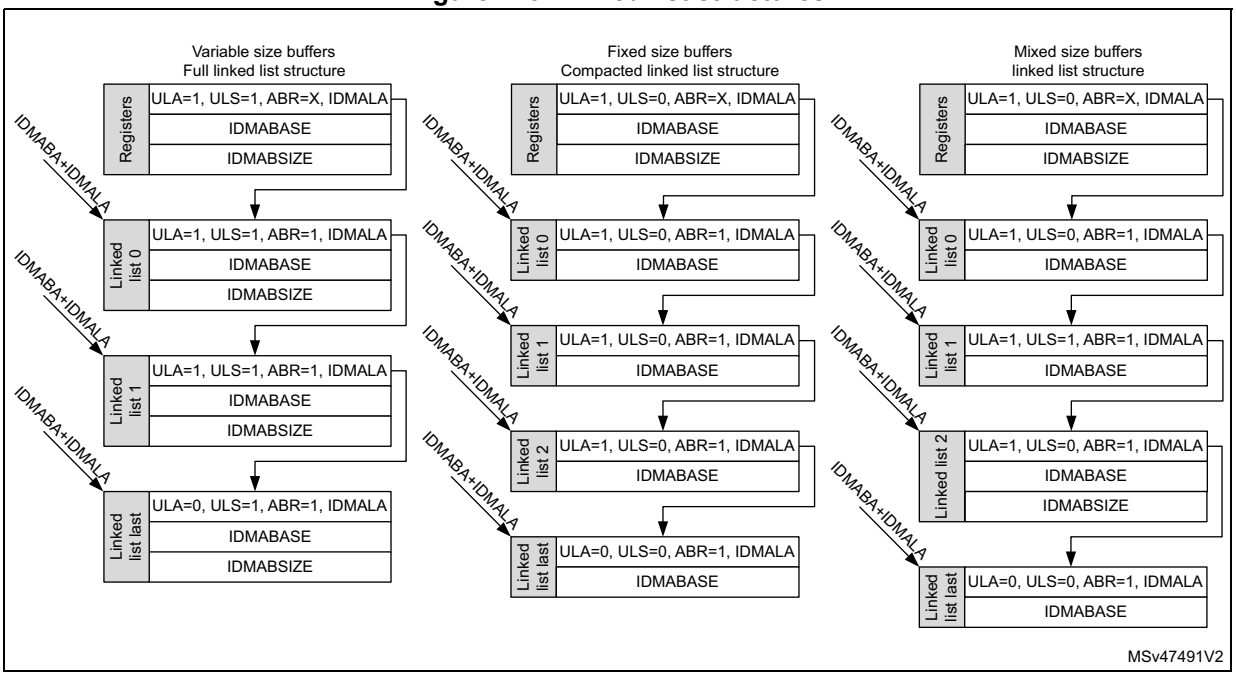
When a IDMA transfer error occurs (see *Section : IDMA transfer error management*) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see *Section : IDMA transfer error management*).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

Figure 173. Linked list structures



There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see *Section : IDMA transfer error management*). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don't care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transfered and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On an IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

24.5.7 AHB and SDMMC_CK clock relation

The AHB must at least have 3x more bandwidth than the SDMMC bus bandwidth: for example, for SDR50 4-bit mode (50 Mbyte/s), the minimum sdmmc_hclk frequency is 37.5 MHz (150 Mbyte/s).

Table 243. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	19.5
e•MMC HS	8	52	39
e•MMC DDR52	8	52	78
e•MMC HS200	8	200	150
SD DS / SDR12	4	25	9.4
SD HS / SDR25	4	50	18.8
SD DDR50	4	50	37.5
SD SDR50	4	100	37.5
SD SDR104	4	208	78

24.6 Card functional description

24.6.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

Table 244. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
Read Wait SDMMC_CK clock stop (START)	X	1	0	1	1
Read Wait SDMMC_CK clock stop (STOP)	X	1	1	1	1
Read Wait SDMMC_D2 signaling (START)	X	0	0	1	1
Read Wait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

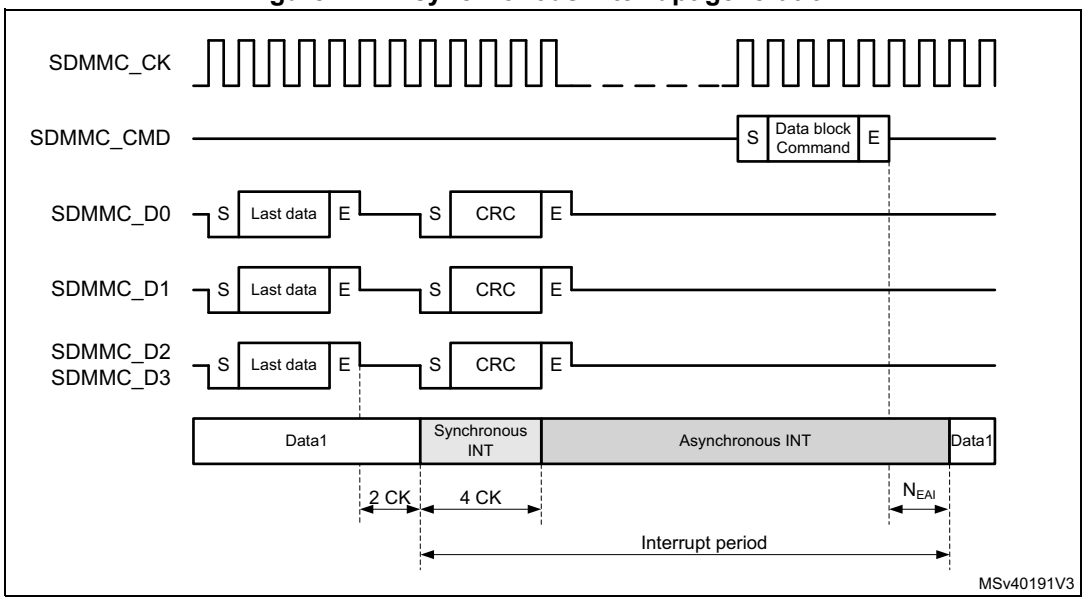
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 244*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 174. Asynchronous interrupt generation



The timing of the interrupt period is depending on the bus speed mode.

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 175. Synchronous interrupt period data read

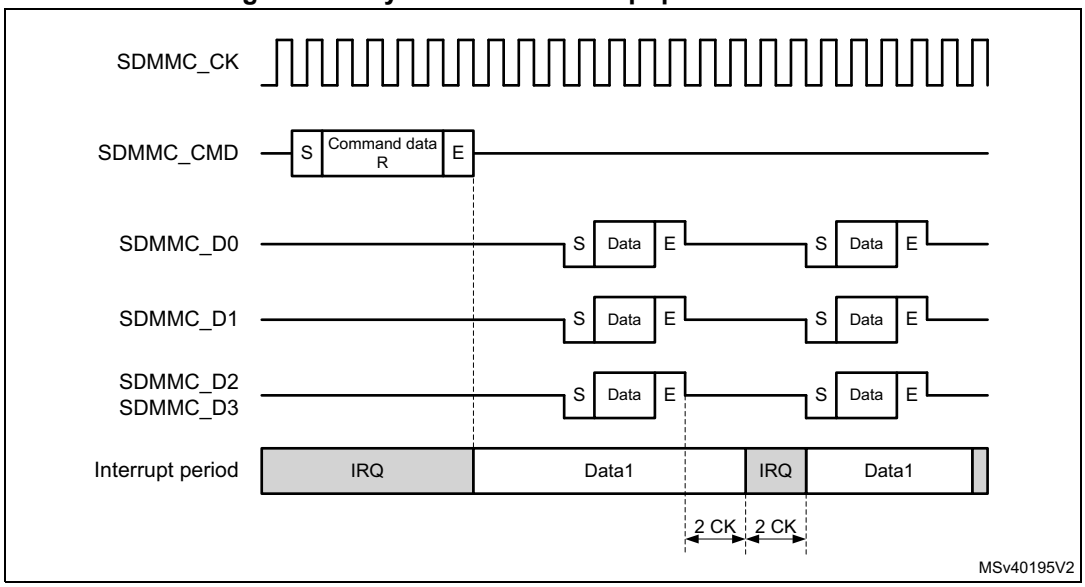
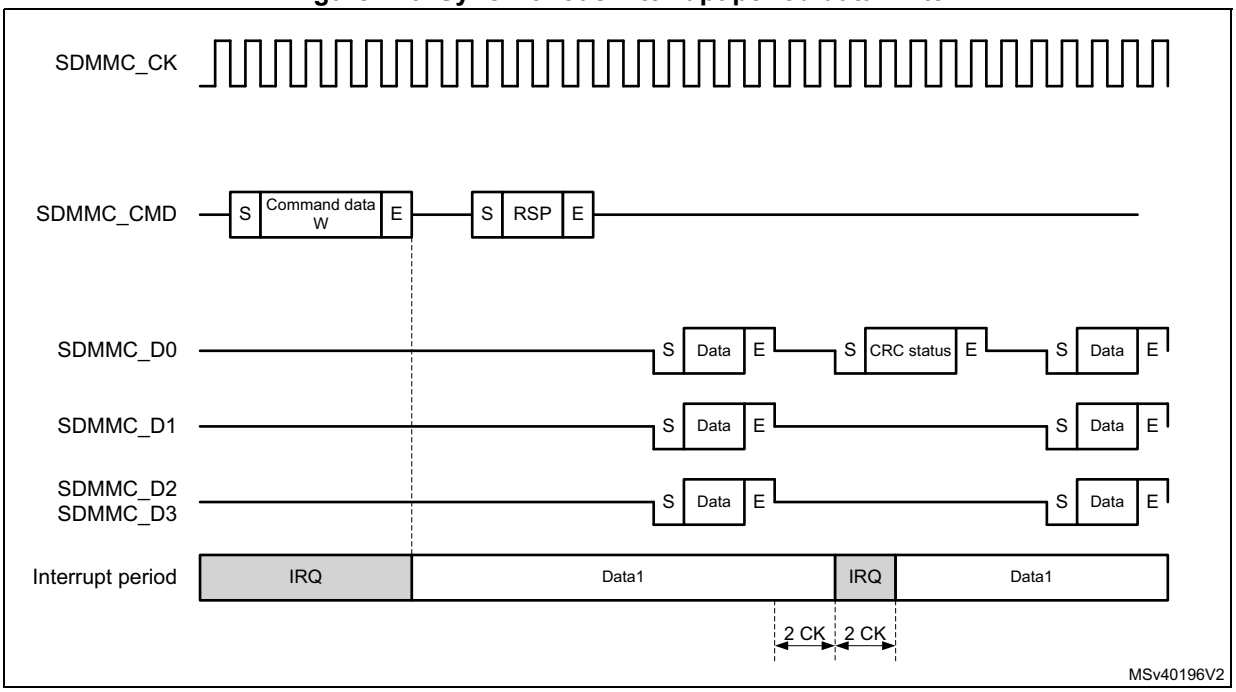


Figure 176. Synchronous interrupt period data write



In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.
- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 177. Asynchronous interrupt period data read

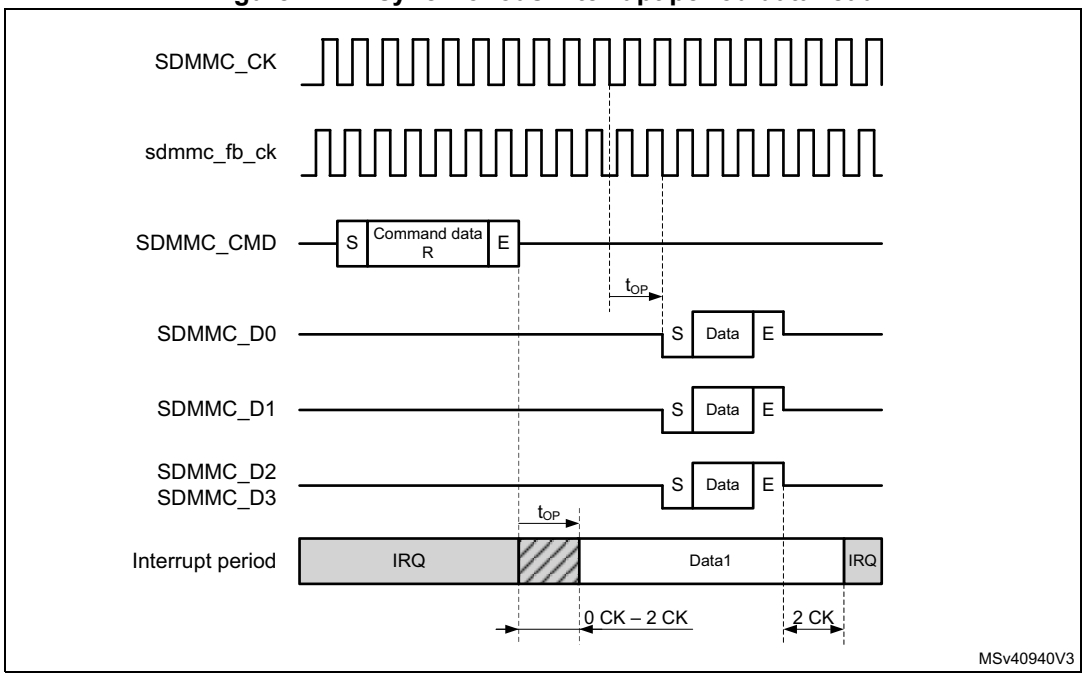
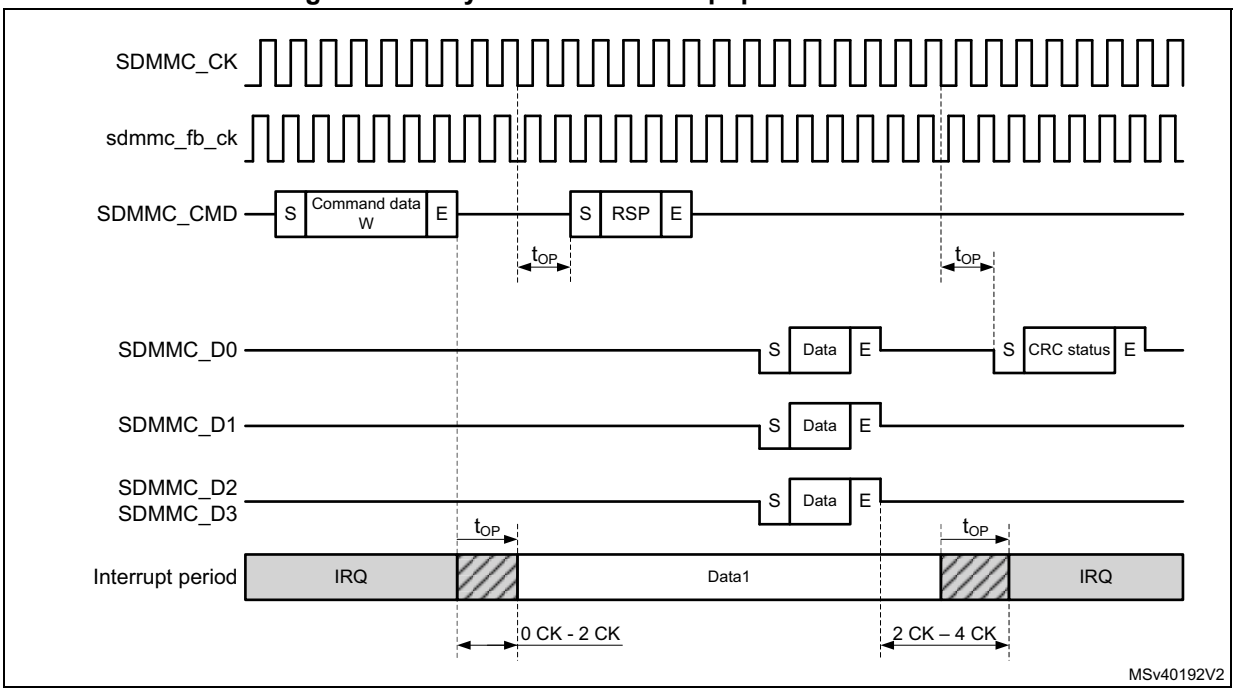


Figure 178. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 245*.

Table 245. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or Read Wait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s).
2. Requests the lower-priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher-priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted (data transfer has suspended).
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

SD I/O Read Wait

There are two methods to pause the data transfer during the block gap:

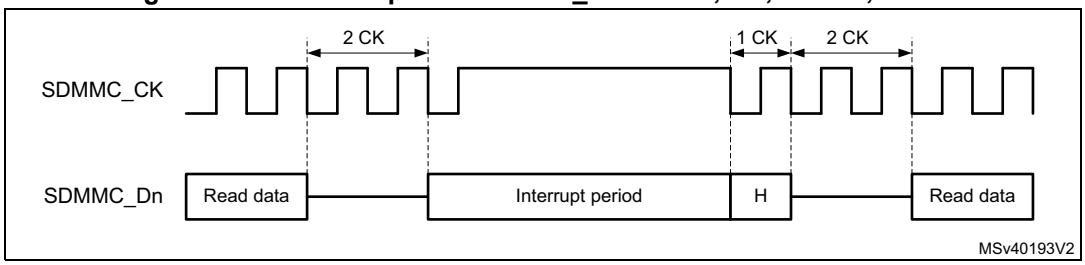
1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

The SDMMC can perform a Read Wait with register settings according *Table 244*.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

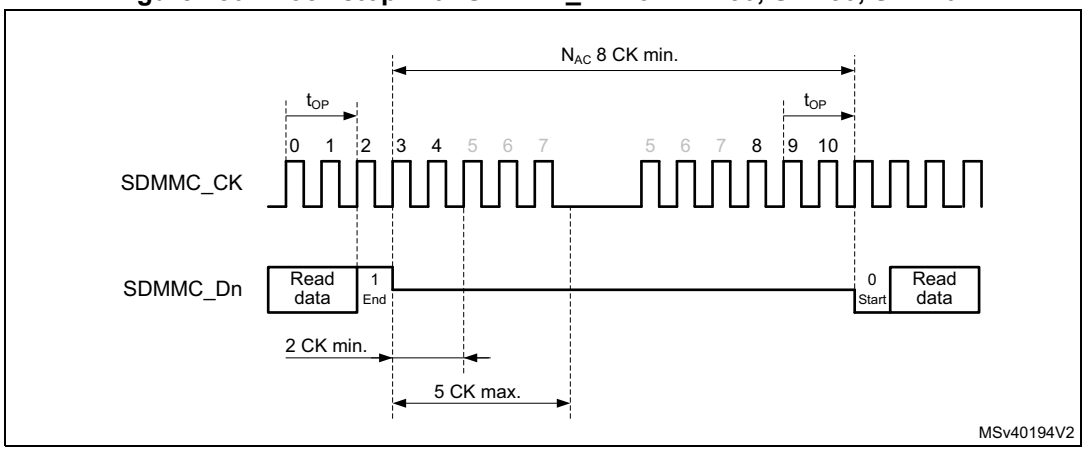
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock (see *Figure 179*).

Figure 179. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see *Figure 180*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 180. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



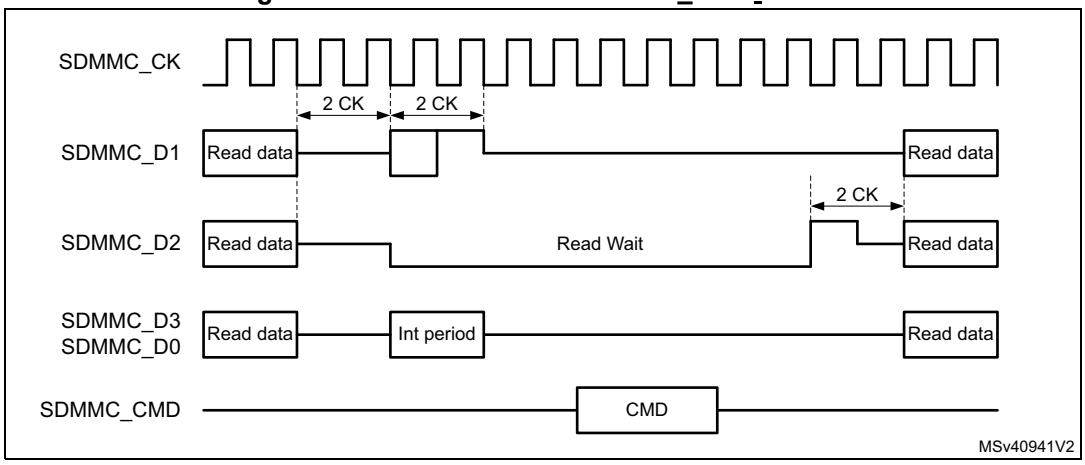
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

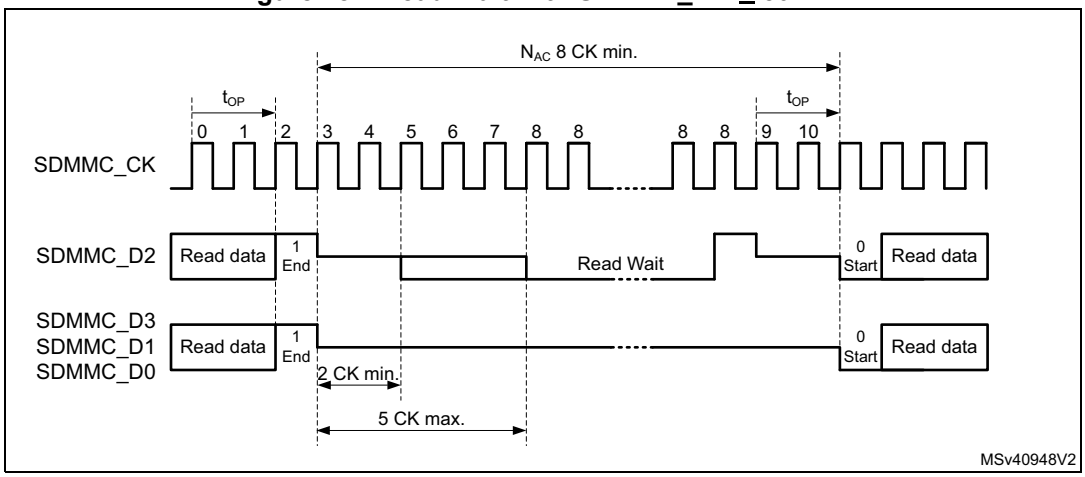
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 181*.

Figure 181. Read Wait with SDMMC_CK < 50 MHz



For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 182*).

Figure 182. Read Wait with SDMMC_CK ≥ 50 MHz



In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

24.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.

Table 246. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

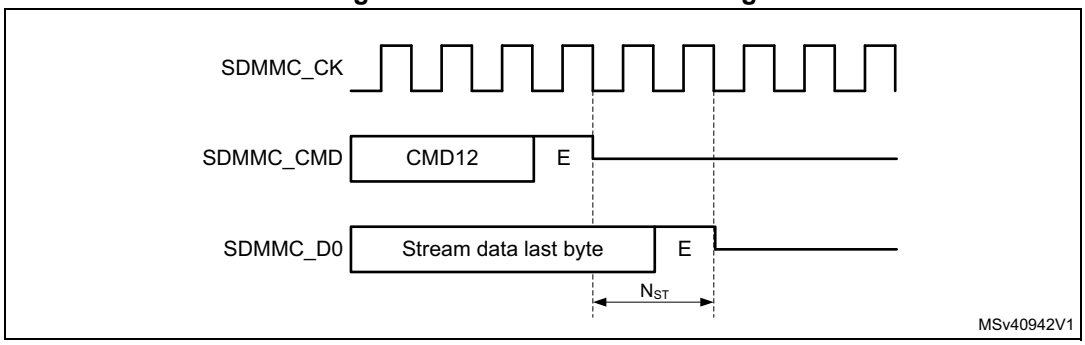
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   – This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   – The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit.
   The card when no data is being sent, does not start sending any new data.
   – The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit.
   The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   – When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes the FIFO.
   – When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   – When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   – When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   – When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   – When DATALENGHT < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 183. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   – CMD12 stops the card sending data.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   – The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   – CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   – CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

24.6.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state.
4. Vcc power supply can be switched off.

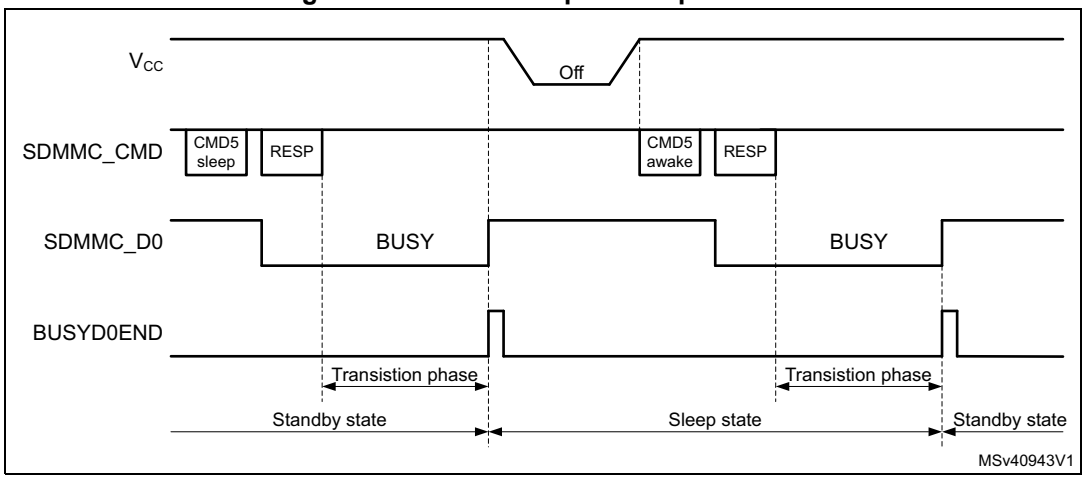
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 184. CMD5 Sleep Awake procedure



24.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   – This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   – Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   – To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

24.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the two boot operation functions:

- Normal boot. (keeping CMD line low)
- Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

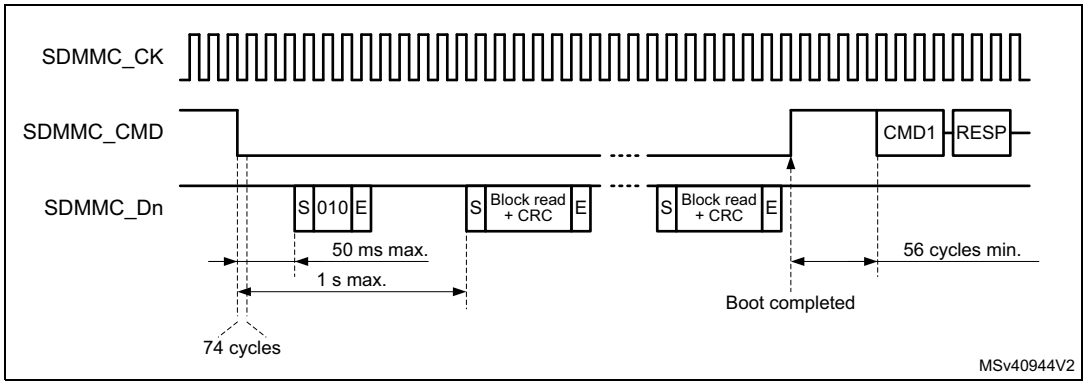
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 185. Normal boot mode operation



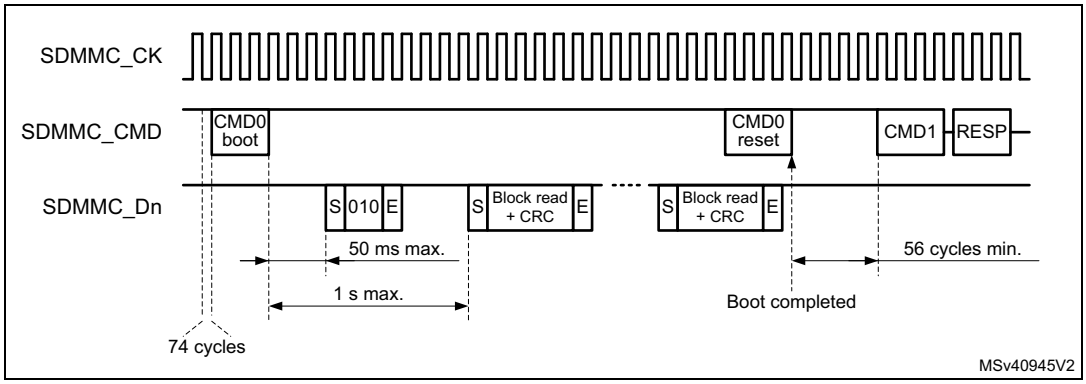
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This causes:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   - If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 186. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

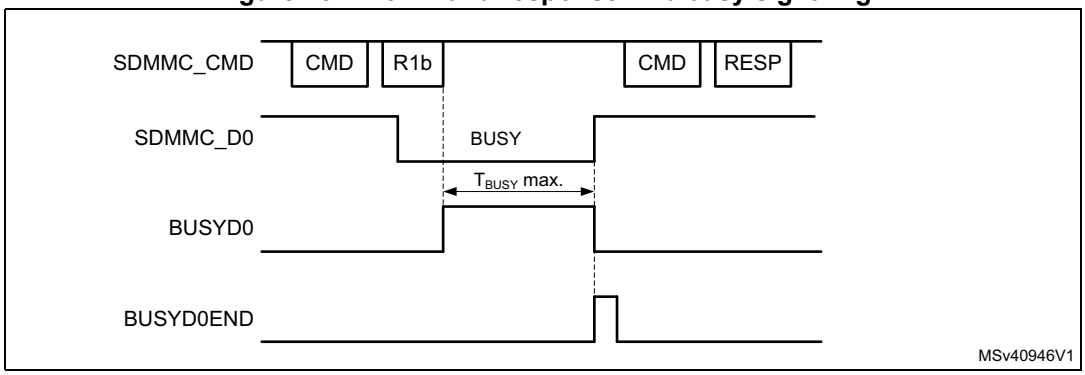
1. Move the SDMMC to power-off state, and reset the card.
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.

9. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
   – if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

24.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 187. Command response R1b busy signaling



The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag.
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  – If BUSYD0 signals not busy, signal busy release to firmware
  – If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

24.6.7 Reset and card cycle power

Reset

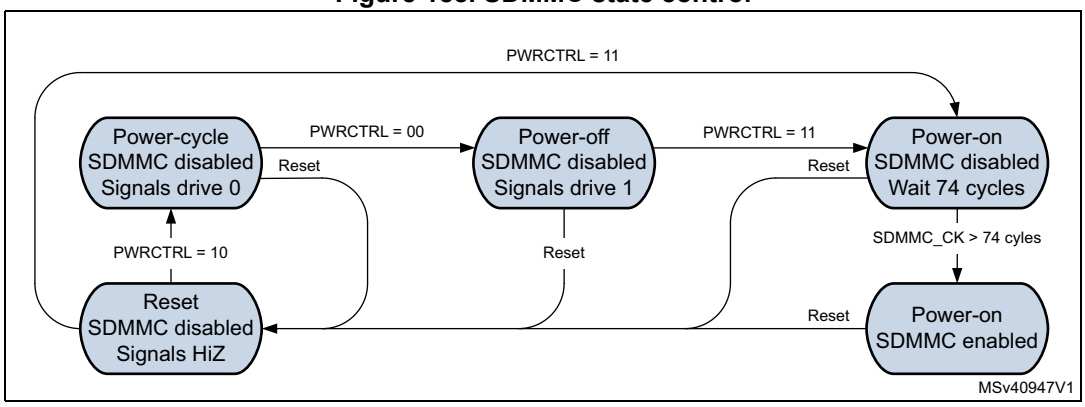
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 188.*.

Figure 188. SDMMC state control

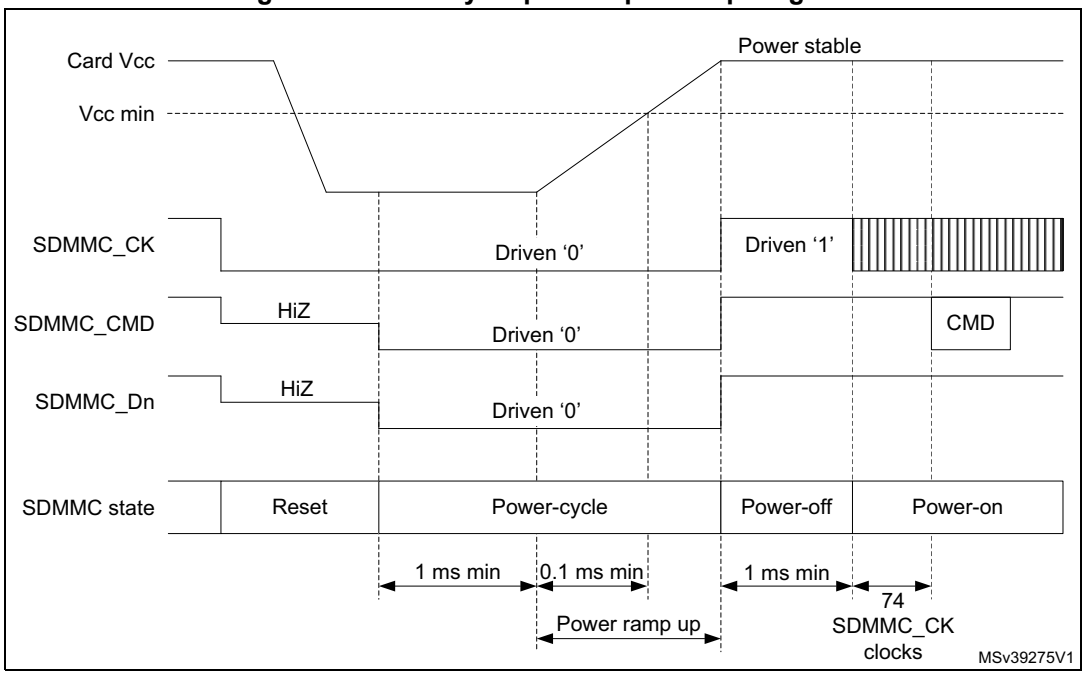


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive 1.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 189. Card cycle power / power up diagram



24.7 Hardware flow control

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

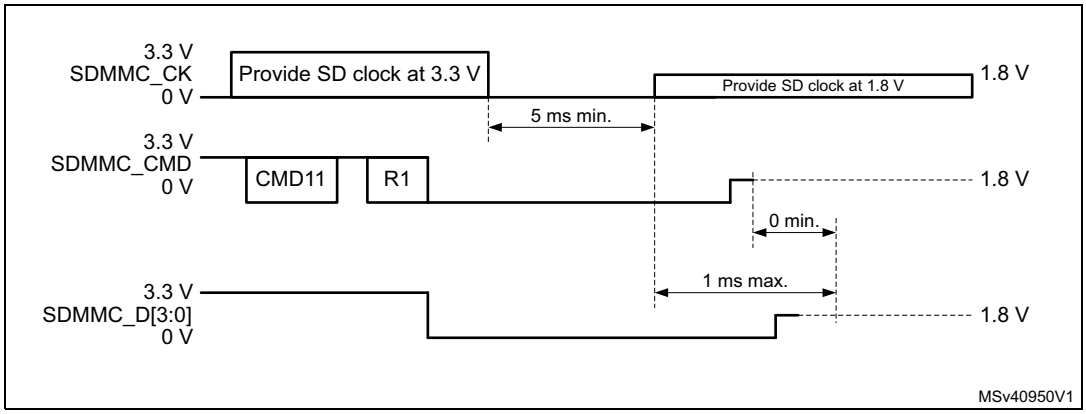
Hardware flow control must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control can not be used.

24.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8 V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch

sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 190. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   - if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.
   - When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].
9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.

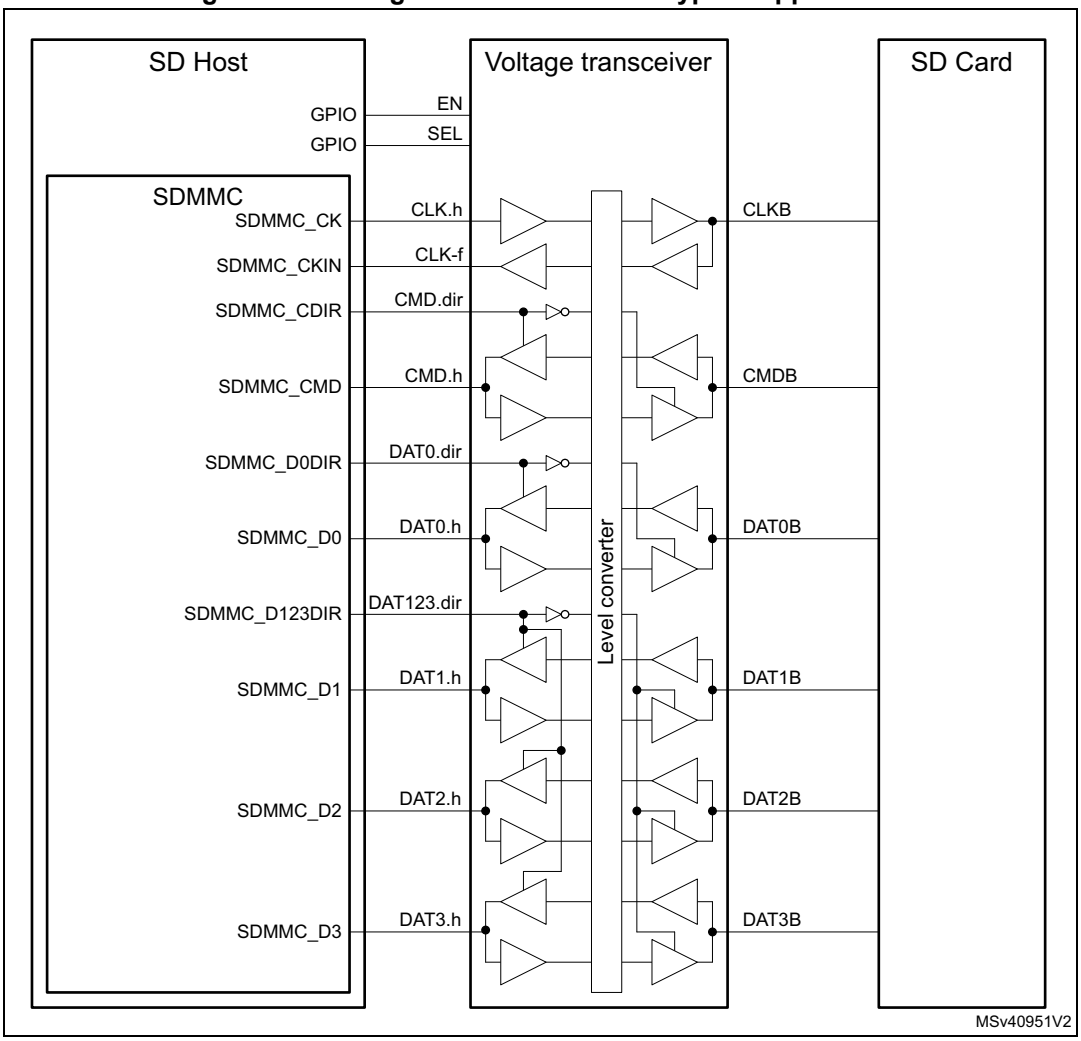
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    – When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by 2^{12} cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by 2^9 cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.

Figure 191. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

- **SDMMC_CKIN** feedback input clock
- **SDMMC_CDIR** I/O direction control for the CMD signal
- **SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal
- **SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

24.9 SDMMC interrupts

Table 247. SDMMC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response CRC fail	CCRCFAIL	CCRCFAILIE	CCRCFAILC	Yes
SDMMC	Data block CRC fail	DCRCFAIL	DCRCFAILIE	DCRCFAILC	Yes
SDMMC	Command response timeout	CTIMEOUT	CTIMEOUTIE	CTIMEOUTC	Yes
SDMMC	Data timeout	DTIMEOUT	DTIMEOUTIE	DTIMEOUTC	Yes
SDMMC	Transmit FIFO underrun	TXUNDERR	TXUNDERRIE	TXUNDERRC	Yes
SDMMC	Receive FIFO overrun	RXOVERR	RXOVERRIE	RXOVERRC	Yes
SDMMC	Command reponse received	CMDREND	CMDRENDIE	CMDRENDC	Yes
SDMMC	Comand sent	CMDSENT	CMDSENTIE	CMDSENTC	Yes
SDMMC	Data transfer ended	DATAEND	DATAENDIE	DATAENDC	Yes
SDMMC	Data transfer hold	DHOLD	DHOLDIE	DHOLDC	Yes
SDMMC	Data block sent or received	DBCKEND	DBCKENDIE	DBCKENDC	Yes
SDMMC	Data transfer aborted	DABORT	DABORTIE	DABORTC	Yes
SDMMC	Transmit FIFO half empty	TXFIFOHE	TXFIFOHEIE	N/A	Yes
SDMMC	Receive FIFO half full	RXFIFOHF	RXFIFOHFIE	N/A	Yes
SDMMC	Transmit FIFO full	TXFIFOF	N/A	N/A	Yes
SDMMC	Receive FIFO full	RXFIFOF	RXFIFOFIE	N/A	Yes
SDMMC	Transmit FIFO empty	TXFIFOE	TXFIFOEIE	N/A	Yes
SDMMC	Receive FIFO empty	RXFIFOE	N/A	N/A	Yes
SDMMC	Command response end of busy	BUSYD0END	BUSYD0ENDIE	BUSYD0ENDC	Yes
SDMMC	SDIO interrupt	SDIOIT	SDIOITIE	SDIOITC	Yes
SDMMC	Boot acknowledgment fail	ACKFAIL	ACKFAILIE	ACKFAILC	Yes
SDMMC	Boot acknowledgment timeout	ACKTIMEOUT	ACKTIMEOUTIE	ACKTIMEOUTC	Yes
SDMMC	Voltage switch timing	VSWEND	VSWENDIE	VSWENDC	Yes
SDMMC	SDMM_CK stopped in voltage switch	CKSTOP	CKSTOPIE	CKSTOPC	Yes
SDMMC	IDMA transfer error	IDMATE	IDMATEIE	IDMATEC	Yes
SDMMC	IDMA buffer transfer complete	IDMABTC	IDMABTCIE	IDMABTCC	Yes

24.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses trigger an AHB bus error.

24.10.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver I/Os driven as output when direction signal is low.

1: Voltage transceiver I/Os driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved (When written 01, PWRCTRL value does not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write is ignored, PWRCTRL value keeps 11.

24.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: sdmmc_fb_ck tuned feedback clock selected as receive clock

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed for selection of SDMMC operating modes

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected

1: SDR50, DDR50, SDR104, HS200 bus speed mode selected

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate must only be selected with clock division > 1 (CLKDIV > 0).

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in *Section 24.10.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: ..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: ..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

24.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains a 32-bit command argument, which is sent to a card as part of a command message.

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

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

24.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS = 0.
CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF = 1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled
1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
0: Normal boot mode procedure selected
1: Alternative boot mode procedure selected

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
If this bit is set, the CPSM is enabled.
When DTEN = 1, no command is transfered nor boot procedure is started. CPSMEN is cleared to 0.
During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: Short response, expect CMDREND flag (No CRC)
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

24.10.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

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

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

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bitfield. Contains the command index of the last command response received.

24.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

These registers contain the status of a card, which is part of the received response.

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

Bits 31:0 **CARDSTATUS[31:0]**: Card status according table below
See *Table 248*.

The card status size is 32 or 128 bits, depending on the response type.

Table 248. Response type and SDMMC_RESPxR registers

Register(1)	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0](2)

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

24.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains the data timeout period, in card bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

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

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

24.10.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

24.10.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

This register controls the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, flushes any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read Wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read Wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.

0: No Read Wait stop

1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8 **RWSTART**: Read Wait start

If this bit is set, Read Wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Define the data block length when the block data transfer mode is selected:

0000: Block length = 2^0 = 1 byte

0001: Block length = 2^1 = 2 bytes

0010: Block length = 2^2 = 4 bytes

0011: Block length = 2^3 = 8 bytes

0100: Block length = 2^4 = 16 bytes

0101: Block length = 2^5 = 32 bytes

0110: Block length = 2^6 = 64 bytes

0111: Block length = 2^7 = 128 bytes

1000: Block length = 2^8 = 256 bytes

1001: Block length = 2^9 = 512 bytes

1010: Block length = 2^{10} = 1024 bytes

1011: Block length = 2^{11} = 2048 bytes

1100: Block length = 2^{12} = 4096 bytes

1101: Block length = 2^{13} = 8192 bytes

1110: Block length = 2^{14} = 16384 bytes

1111: Reserved

When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transfered.)

When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

00: Block data transfer ending on block count.

01: SDIO multibyte data transfer.

10: *e*•MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)

11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: From host to card.

1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by hardware when data transfer completes.

This bit must only be used to transfer data when no associated data transfer command is used (must not be used with SD or *e*•MMC cards).

0: Do not start data transfer without CPSM data transfer command.

1: Start data transfer without CPSM data transfer command.

24.10.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

This register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and

when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

24.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
0: card signals not busy on SDMMC_D0.
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state
or
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.

24.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

This register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOPC	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	D ABORT C	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

24.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			rw		rw	rw	rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVER RIE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by voltage switch clock stopped.
0: Voltage switch clock stopped interrupt disabled
1: Voltage switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment fail.
0: Acknowledgment fail interrupt disabled
1: Acknowledgment fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO mode interrupt received interrupt disabled
1: SDIO mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

24.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

24.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.

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

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

Bit 1 **IDMABMODE**: Buffer mode selection
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: Single buffer mode.
1: Linked list mode.

Bit 0 **IDMAEN**: IDMA enable
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: IDMA disabled
1: IDMA enabled

24.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

This register contains the buffer size when in linked list configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11]
															rw

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

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:5 **IDMABNDT[11:0]**: Number of bytes per buffer
This 12-bit value must be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.
Example: IDMABNDT = 0x001: buffer size = 8 words = 32 bytes.
Example: IDMABNDT = 0x800: buffer size = 16384 words = 64 Kbyte.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 4:0 Reserved, must be kept at reset value.

24.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

This register contains the memory buffer base address in single buffer configuration and linked list configuration.

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

Bits 31:0 **IDMABASE[31:0]**: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).

24.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMALAR is not to be updated, last linked list item.
1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMABSIZER is not to be updated from next linked list table.
1: SDMMC_IDMABSIZER is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
This bit is not taken into account when starting the first linked list buffer from the software programmed register information. ABR is only taken into account on subsequent loaded linked list items.
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be generated).
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset
Linked list item offset pointer to the base of the next linked list item structure.
Linked list item base address is IDMABA + IDMALA.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

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

24.10.19 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR)

Address offset: 0x068

Reset value: 0x0000 0000

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

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address
Linked list memory base pointer.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

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

24.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

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

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).

The FIFO data occupies 16 entries of 32-bit words.

24.10.21 SDMMC register map

Table 249. SDMMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SDMMC_ POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	SDMMC_ CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ ARGR	CMDARG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SDMMC_ CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																

Table 249. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SDMMC_ RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	SDMMC_ RESP1R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SDMMC_ RESP2R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SDMMC_ RESP3R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SDMMC_ RESP4R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SDMMC_ DTIMER	DATATIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SDMMC_ DLENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SDMMC_ DCTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SDMMC_ DCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SDMMC_ STAR	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SDMMC_ ICR	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SDMMC_ MASKR	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0

Table 249. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SDMMC_ ACKTIMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044 - 0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x050	SDMMC_ IDMACTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
	Reset value																															0	0
0x054	SDMMC_ IDMABSIZER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[11:0]												Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0					
0x058	SDMMC_ IDMABASER	IDMABASE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C - 0x060	Reserved	Res.																															
0x064	SDMMC_ IDMALAR	ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMALA[13:0]														Res.	Res.
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0			
0x068	SDMMC_ IDMABAR	IDMABA[29:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x06C - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080 + 0x04 * x, (x=0..15)	SDMMC_ FIFORx	FIFODATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

25 Delay block (DLYB)

25.1 DLYB introduction

The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC or Octo-SPI interface.

The delay is voltage- and temperature-dependent, that may require the application to re-configure and recenter the output clock phase with the receive data.

25.2 DLYB main features

The delay block has the following features:

- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

25.3 DLYB implementation

Table 250. STM32H5 features

DLYB associated peripheral	DLYBOS1	DLYBSD1	DLYBSD2(1)
OCTOSPI1	X	-	-
SDMMC1	-	X	-
SDMMC2	-	-	X

1. Only available on STM32H56 and STM32H57 devices.

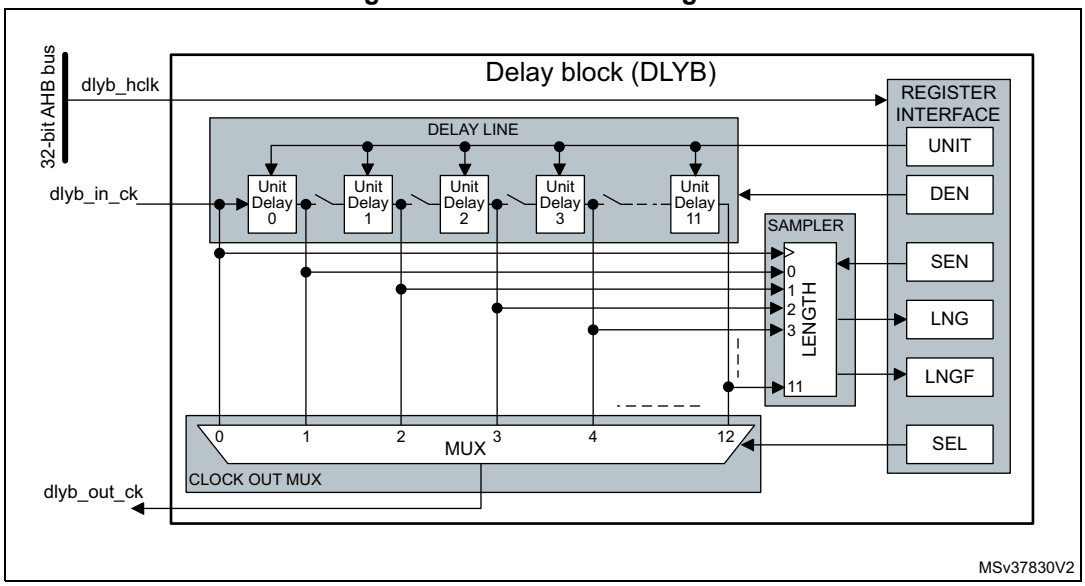
25.4 DLYB functional description

25.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 192. DLYB block diagram



25.4.2 DLYB pins and internal signals

Table 251 lists the DLYB internal signals.

Table 251. DLYB internal input/output signals

Signal name	Signal type	Description
dlyb_hclk	Digital input	Delay block register interface clock
dlyb_in_ck	Digital input	Delay block input clock
dlyb_out_ck	Digital output	Delay block output clock

25.4.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 252. Delay block control

DEN	SEN	UNIT	SEL	LNG	LNGF	Output clock
0	0	Don't care	Don't care	Don't care	Don't care	Enabled (= Input clock)
x	1	Unit delay	Output clock phase	Length	Length flag	Disabled
1	0	Unit delay(1)	Output clock phase(2)	Don't care	Don't care	Enabled (= selected phase)

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.

25.4.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].

   If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

25.4.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

Octo-SPI use case:

The delay block is used in conjunction with Octo-SPI interface to allow shifting the input data sampling signal. This sampling signal can be the feedback clock or the data strobe (DQS) signal, which is delivered by certain type of devices. Note that in case DQS is used, the calibration procedure must be performed beforehand with a free running clock, as DQS is a discontinuous signal.

In case of SDR (single data rate) mode the user must typically shift the sampling signal by half period, so that the sampling edges are positioned in the middle of the valid data phase.

In case of DDR (dual data rate) mode, for which data are transitioning at start and middle of period, typical value must be close to N/4, once the calibration is completed.

In case of high frequencies and tight timing constraints, the delay setting granularity (10) might be too coarse. Since in most cases it is not necessary to have a possible delay value covering the whole sampling clock period, the "Unit" value can be overridden by application in order to improve the accuracy of the sampling edge position (example: providing a twice as small "Unit" gives twice better timing accuracy. The counterpart being that the maximum possible delay is divided by 2).

SDMMC use case:

The delay block is used in conjunction with SDMMC interface variable delay. For correct sampling point tuning the delay value must cover a whole SDMMC_CK clock period. After having tuned the delay line length the individual delays are used in the sampling point tuning to find the optimal sampling point.

25.5 DLYB registers

All registers can be accessed in word, half-word and byte access.

25.5.1 DLYB control register (DLYB_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 1 **SEN:** Sampler length enable bit

0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.

1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN:** Delay block enable bit

0: DLYB disabled.

1: DLYB enabled.

25.5.2 DLYB configuration register (DLYB_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bit 31 **LNGF:** Length valid flag

This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.

0: Length value in LNG is not valid.

1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **LNG[11:0]:** Delay line length value

These bits reflect the 12 unit delay values sampled at the rising edge of the input clock. The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **UNIT[6:0]:** Delay of a unit delay cell.

These bits can only be written when SEN = 1.

Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]:** Phase for the output clock.

These bits can only be written when SEN = 1.

Output clock phase = input clock + SEL[3:0] x unit delay

25.5.3 DLYB register map

Table 253. DLYB register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DLYB_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEN	DEN
	Reset value																															0	0
0x004	DLYB_CFGR	LNGF	Res.	Res.	Res.	LNG[11:0]												Res.	UNIT[6:0]							Res.	Res.	Res.	Res.	SEL[3:0]			
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

26 Analog-to-digital converters (ADC1/2)

26.1 ADC introduction

This section describes the implementation of up to 2 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

26.2 ADC main features

- High-performance features
  - Up to 2 ADCs which can operate in dual mode:
    ADC1 is connected to 18 external channels and to 2 internal channels
    ADC2 is connected to 18 external channels and to 2 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Four dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency

  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)
- Oversampler
  - 16-bit data register
  - Oversampling ratio adjustable from 2 to 256x
  - Programmable data shift up to 8 bits
- Data preconditioning
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC):

    Up to 6 fast channels from GPIO pads

    Up to 12 slow channels from GPIO pads
  - 1 channel for the internal temperature sensor (V_{SENSE})
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for monitoring the external VBAT power supply pin
  - 1 channel for monitoring the internal V_{DDCORE} supply
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
  - Watchdog can perform filtering to ignore out-of-range data
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 193 shows the block diagram of one ADC.

26.3 ADC implementation

Table 254. ADC features

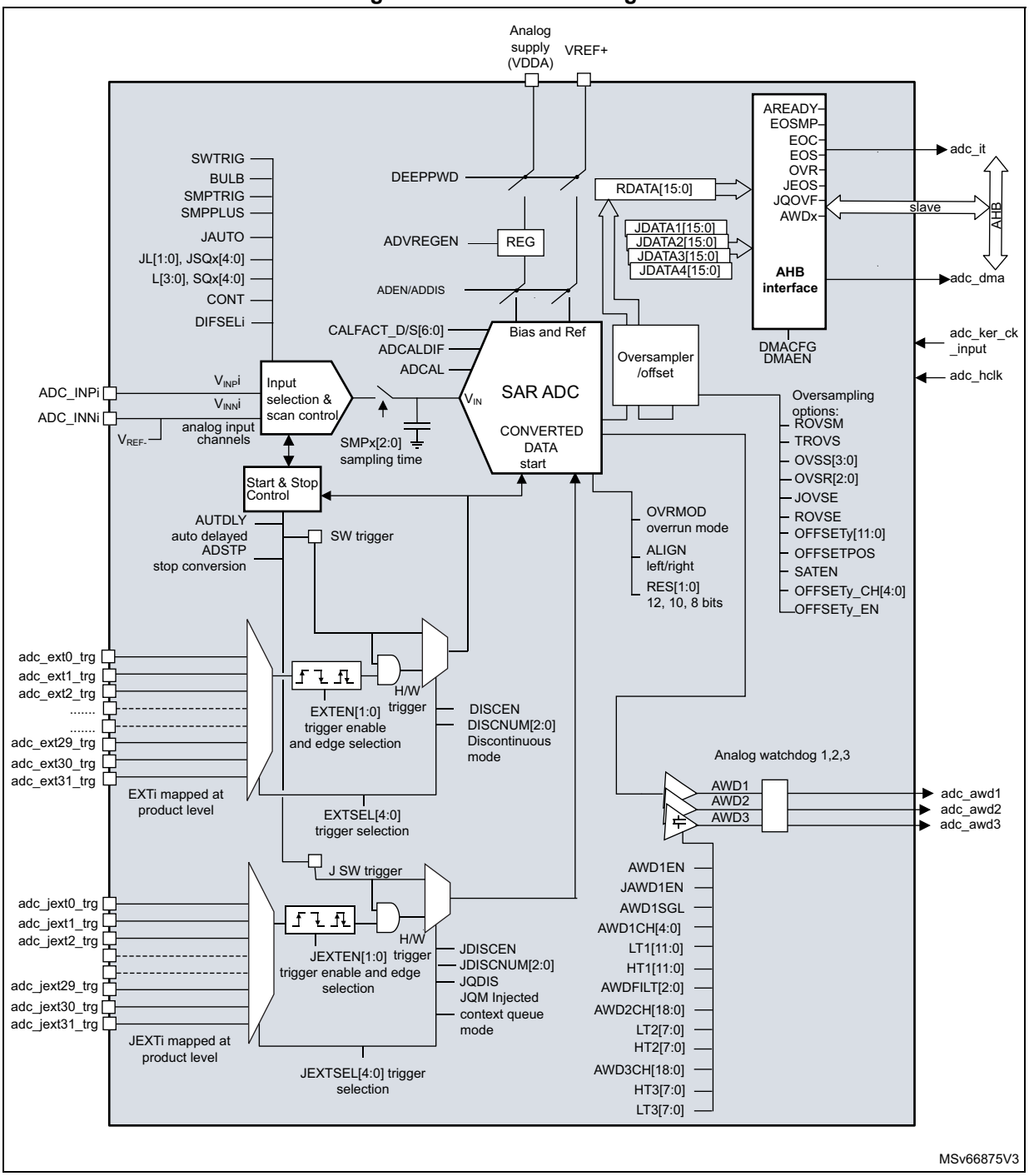
ADC modes/features	ADC1	ADC2
Resolution	12 bit	
Maximum sampling speed	5 Msps (12-bit resolution)	
Dual mode operation	X	
Hardware offset calibration	X	
Hardware linearity calibration	-	
Single-end input	X	
Differential input	X	
Injected channel conversion	X	
Oversampling	up to x256	
Data register	16 bits	
Data register FIFO depth	3 stages	
DMA support	X	
Parallel data output to ADF	-	
Offset compensation	X	
Gain compensation	-	
Number of analog watchdog	3	
Option register	X	

26.4 ADC functional description

26.4.1 ADC block diagram

Figure 193 shows the ADC block diagram and *Table 255* gives the ADC pin description.

Figure 193. ADC block diagram



26.4.2 ADC pins and internal signals

Table 255. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply and positive reference voltage for the ADC
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VREF−	Input, analog reference negative	The lower/negative reference voltage for the ADC. V_{REF-} is internally connected to V_{SSA}
ADC1/2_INNi/INPi	Negative/positive external analog input signals	20 negative/positive external analog input channels (refer to *Section 26.4.4: ADC connectivity* for details)

Table 256. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}i$	Positive analog input channels	Positive internal analog input channels connected either to ADC1/2_INP*i* external channels or to internal channels.
$V_{INN}i$	Negative analog input channels	Negative internal analog input channels connected either to ADC1/2_INN*i* external channels or to internal channels
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.
adc_awdx	Output	Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)
adc_ker_ck_input	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request

Table 257. ADC interconnection

Signal name	Source/destination
ADC1 V_{INP}[16]	V_{SENSE} (internal temperature sensor output voltage).
ADC1 V_{INP}[17]	V_{REFINT} (output voltage from internal reference voltage).
ADC2 V_{INP}[16]	V_{BAT}/4 (VBAT pin input voltage divided by 4).
ADC2 V_{INP}[17]	V_{DDCORE} (internal digital core voltage).
adc_ext_trg0	tim1_oc1

Table 257. ADC interconnection (continued)

Signal name	Source/destination
adc_ext_trg1	tim1_oc2
adc_ext_trg2	tim1_oc3
adc_ext_trg3	tim2_oc2
adc_ext_trg4	tim3_trgo
adc_ext_trg5	tim4_oc4
adc_ext_trg6	exti11
adc_ext_trg7	tim8_trgo
adc_ext_trg8	tim8_trgo2
adc_ext_trg9	tim1_trgo
adc_ext_trg10	tim1_trgo2
adc_ext_trg11	tim2_trgo
adc_ext_trg12	tim4_trgo
adc_ext_trg13	tim6_trgo
adc_ext_trg14	tim15_trgo
adc_ext_trg15	tim3_oc4
adc_ext_trg16	exti15
adc_ext_trg17	reserved
adc_ext_trg18	lptim1_ch1
adc_ext_trg19	lptim2_ch1
adc_ext_trg20	reserved
adc_ext_trg21	reserved
adc_ext_trg22	reserved
adc_ext_trg23	reserved
adc_ext_trg24	reserved
adc_ext_trg25	reserved
adc_ext_trg26	reserved
adc_ext_trg27	reserved
adc_ext_trg28	reserved
adc_ext_trg29	reserved
adc_ext_trg30	reserved
adc_ext_trg31	reserved
adc_jext_trg0	tim1_trgo
adc_jext_trg1	tim1_oc4
adc_jext_trg2	tim2_trgo
adc_jext_trg3	tim2_oc1

Table 257. ADC interconnection (continued)

Signal name	Source/destination
adc_jext_trg4	tim3_oc4
adc_jext_trg5	tim4_trgo
adc_jext_trg6	exti15
adc_jext_trg7	tim8_oc4
adc_jext_trg8	tim1_trgo2
adc_jext_trg9	tim8_trgo
adc_jext_trg10	tim8_trgo2
adc_jext_trg11	tim3_oc3
adc_jext_trg12	tim3_trgo
adc_jext_trg13	tim3_oc1
adc_jext_trg14	tim6_trgo
adc_jext_trg15	tim15_trgo
adc_jext_trg16	reserved
adc_jext_trg17	reserved
adc_jext_trg18	lptim1_ch1
adc_jext_trg19	lptim2_ch1
adc_jext_trg20	reserved
adc_jext_trg21	reserved
adc_jext_trg22	reserved
adc_jext_trg23	reserved
adc_jext_trg24	reserved
adc_jext_trg25	reserved
adc_jext_trg26	reserved
adc_jext_trg27	reserved
adc_jext_trg28	reserved
adc_jext_trg29	reserved
adc_jext_trg30	reserved
adc_jext_trg31	reserved

26.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The ADC input clock can be selected between two different clock sources (see *Figure 194: ADC clock scheme*):

1. The ADC clock can be a specific clock source (adc_ker_ck_input), independent and asynchronous with the AHB clock.

   Refer to section *Reset and clock control (RCC)* for more information on how to generate the ADC dedicated clock. To select this scheme, CKMODE[1:0] bits of ADC_CCR register must be set to 00.

2. The ADC clock can be derived from the AHB clock interface divided by a programmable factor of 1, 2 or 4. To select this scheme, CKMODE[1:0] bits of ADC_CCR must be different from 00. The programmable divider factor can be configured through to CKMODE[1:0] bits of ADC_CCR.

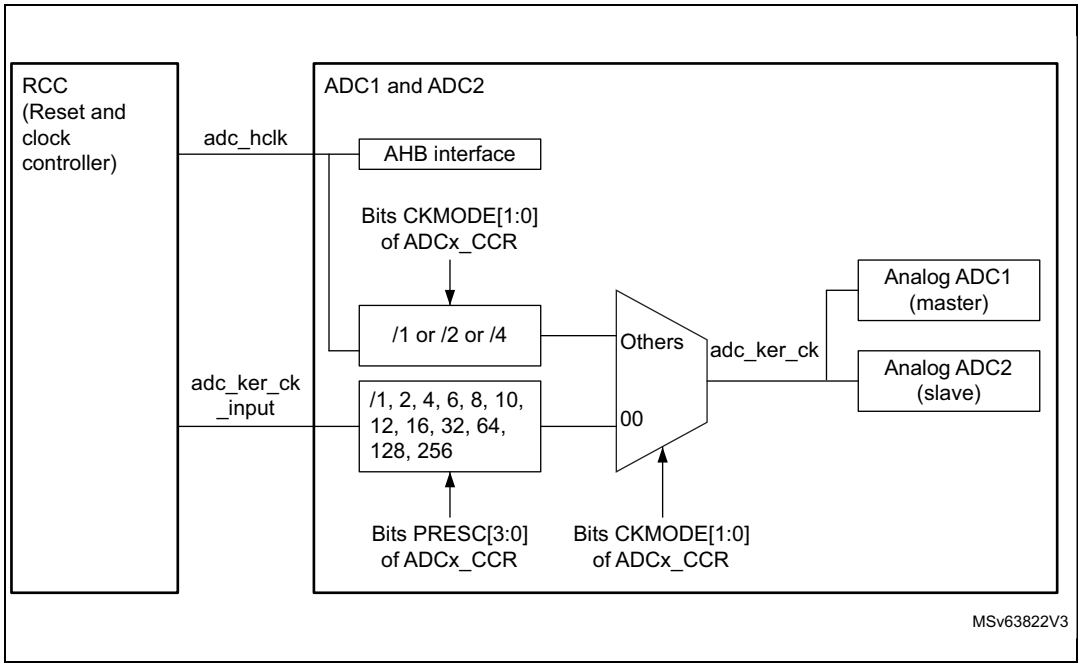
   The prescaling factor of 1 (CKMODE[1:0] = 01) can be used only if the AHB prescaler is set to 1 (HPRE[3:0] = 0xxx in the RCC_CFGR register).

Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256, using the prescaler configured with bits PRESC[3:0] in the ADC_CCR register.

Option **2** has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock is configured through CKMODE[1:0] bits must be compliant with the operating frequency specified in the device datasheet.

Figure 194. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{adc_hclk} \geq F_{ADC} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{adc_hclk} \geq F_{ADC} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- $F_{adc_hclk} \geq F_{ADC} / 2$ if there are some channels with resolutions equal to 6-bit

Constraints between ADC clocks

When several ADC interfaces are used simultaneously, it is mandatory to use the same clock source from the RCC block without prescaler ratio for all ADC interfaces.

26.4.4 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described below.

Figure 195. ADC1 connectivity

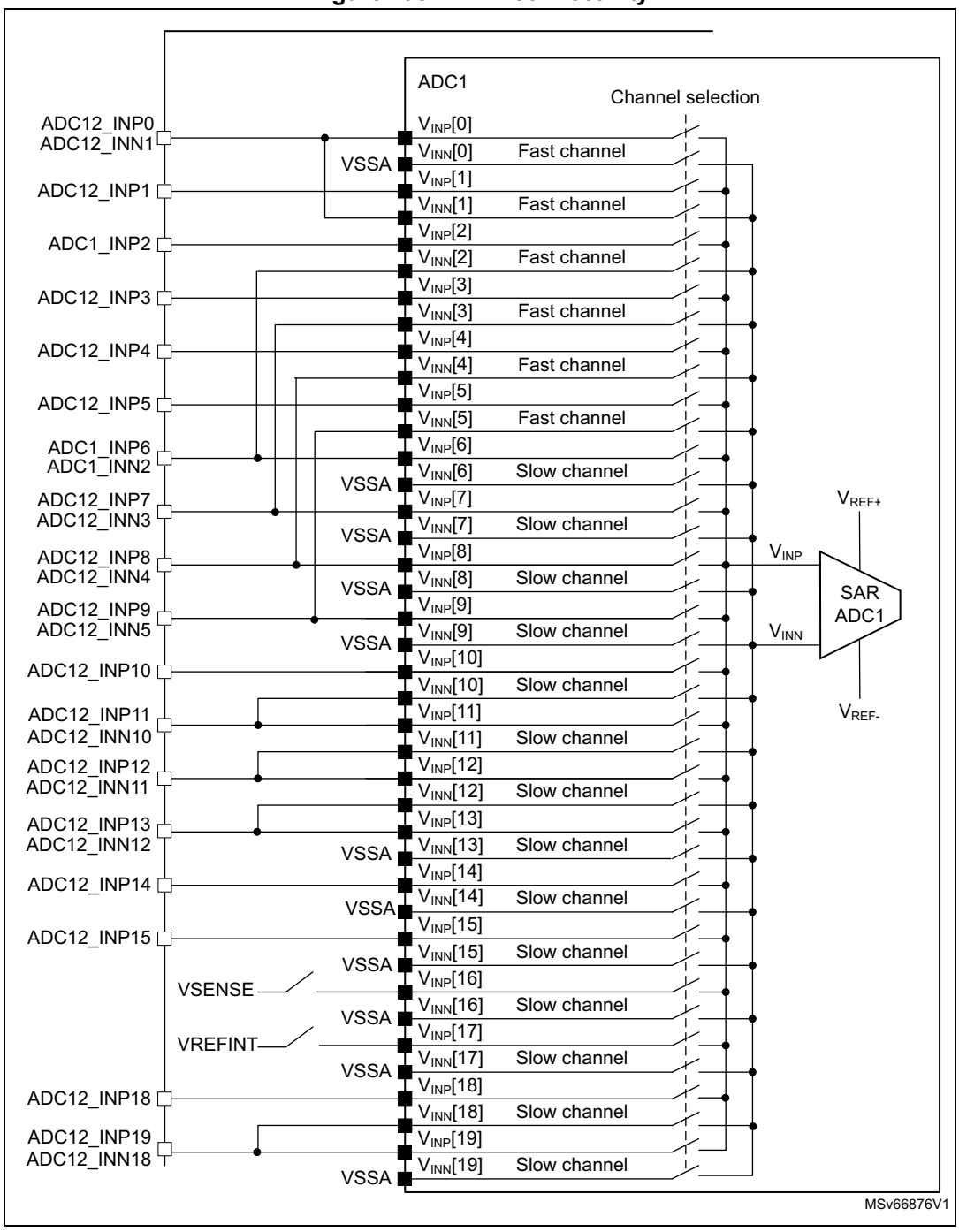
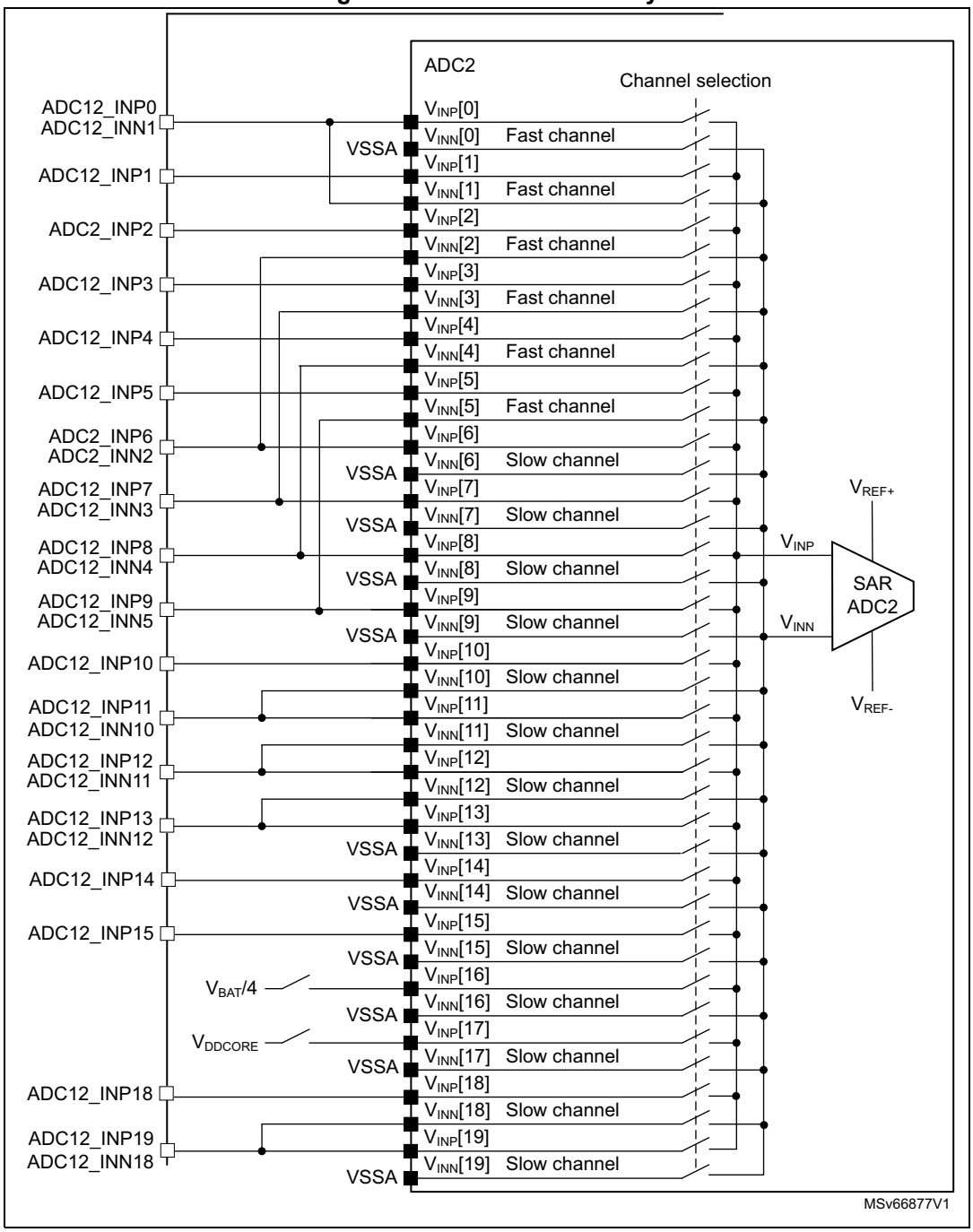


Figure 196. ADC2 connectivity



26.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

26.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPPWD is 1 in the ADC_CR register).

To start ADC operations, it is first needed to exit Deep-power-down mode by setting bit DEEPPWD = 0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN = 1 into ADC_CR register. The software must wait for the startup time of the ADC voltage regulator ($T_{ADCVREG_STUP}$) before launching a calibration or enabling the ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, refer to device datasheet for $T_{ADCVREG_STUP}$ parameter.

When ADC operations are complete, the ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPPWD = 1 into ADC_CR register. This is particularly interesting before entering Stop mode.

Note: Writing DEEPPWD = 1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.

When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.

In ADC Deep-power-down mode (DEEPPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to *Section 26.4.8: Calibration (ADCAL, ADCALDIF, ADC_CALFACT)*).

26.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and V_{REF-} (negative input).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

The output data for the differential mode is an unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC_Full_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When ADC is configured as differential mode, both inputs should be biased at (V_{REF+}) /2 voltage.

The input signals are supposed to be differential (common mode voltage should be fixed).

Internal channels (such as V_{REFINT} and V_{SENSE}) are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to *Section 26.4.4: ADC connectivity*.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted. Some channels are shared between ADC1/ADC2: this can make the channel on the other ADC unusable. Only exception is interleaved mode for ADC master and the slave.

26.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIF = 0 before launching a calibration which is applied for single-ended input conversions.
- Write ADCALDIF = 1 before launching a calibration which is applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). ADCAL bit stays at 1 during all the

calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration)

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

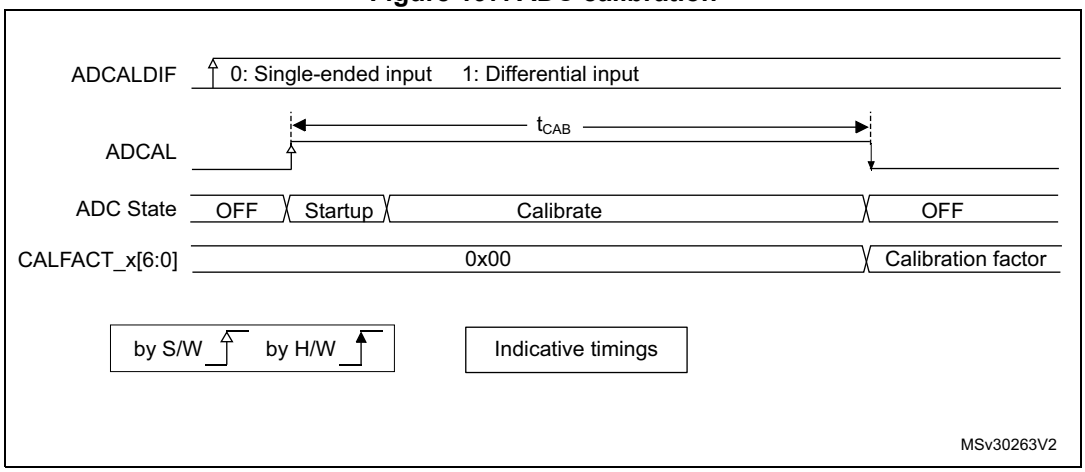
The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or V_{BAT} mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when V_{REF+} voltage changed more than 10%.

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL = 1.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC_CALFACT register.

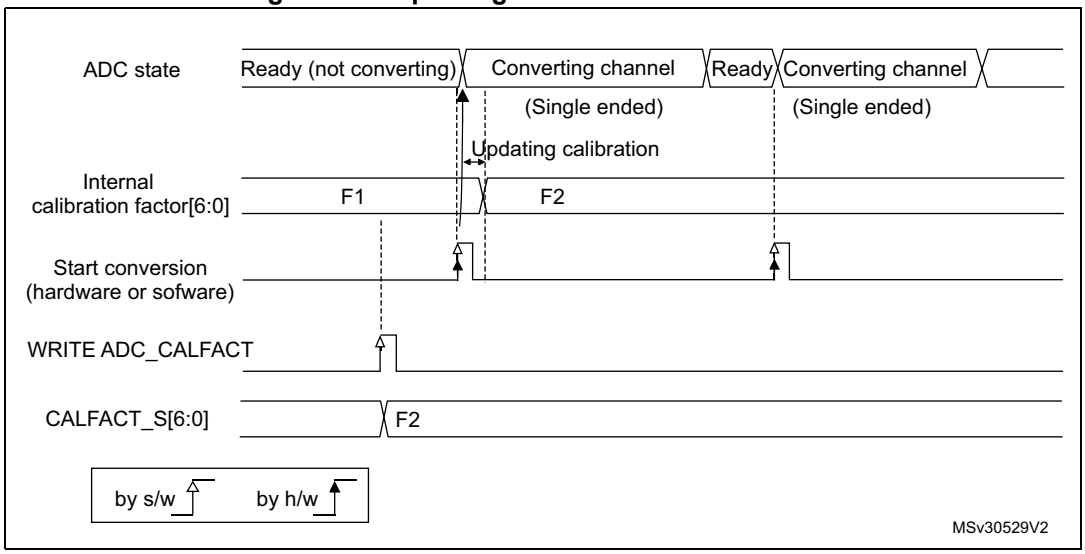
Figure 197. ADC calibration



Software procedure to reinject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Figure 198. Updating the ADC calibration factor

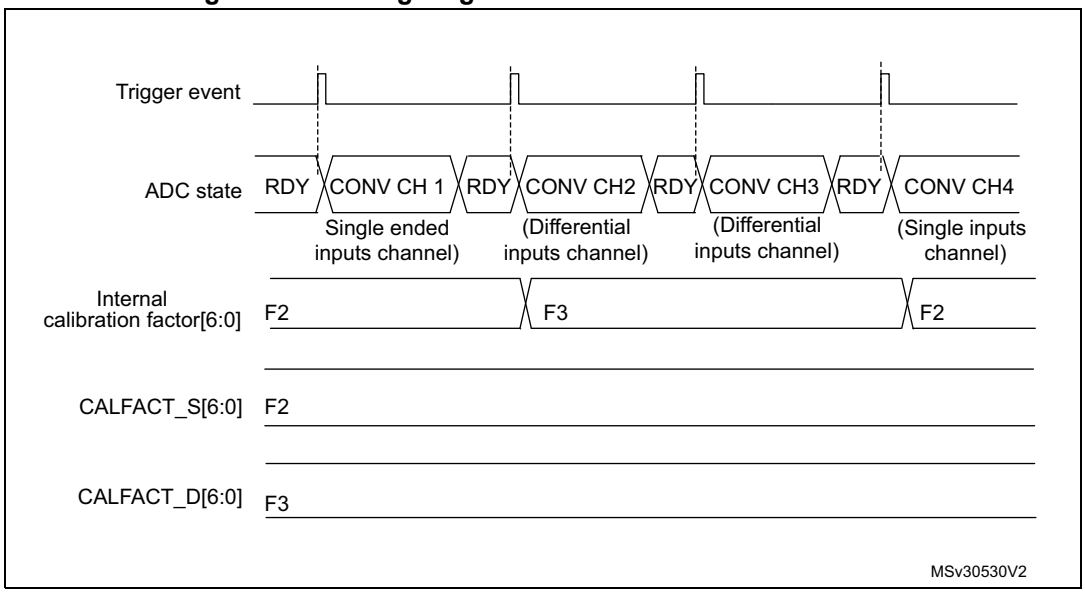


Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.

Figure 199. Mixing single-ended and differential channels



26.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in *Section 26.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*).

Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of t_{STAB} before it starts converting accurately, as shown in *Figure 200*. Two control bits enable or disable the ADC:

- ADEN = 1 enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 (refer to *Section 26.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART = 1 or when an external injected trigger event occurs, if injected triggers are enabled.

Software procedure to enable the ADC

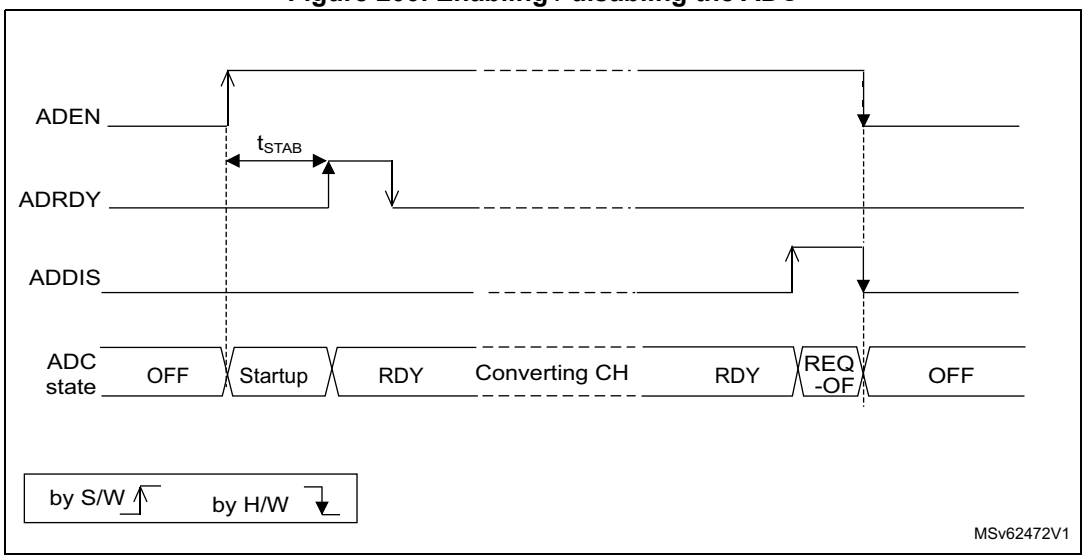
1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN = 1.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

Caution: ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS = 1.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 200. Enabling / disabling the ADC



26.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).
- ADC_TRy registers can be modified when an analog-to-digital conversion is ongoing (refer to *Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)* for details).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).

The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN = 1). Refer to *Section 26.6.16: ADC injected sequence register (ADC_JSQR)* for additional details.

Note: There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN = 0 as well as all the bits of ADC_CR register).

26.4.11 Channel selection (ADC_SQRy, ADC_JSQR)

The ADC features up to 20 multiplexed channels per ADC, out of which:

- Up to 18 analog inputs coming from GPIO pads (ADC_INP/INN[i]) depending on the products, not all of them are available on GPIO pads.
- Up to 4 internal channels (refer to *Section 26.4.4: ADC connectivity* for details)

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, VBATEN or TSEN in the ADC_CCR registers.

Refer to *Table ADC interconnection* in *Section 26.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to external ADC pins or internal signals.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC1/2_INP/INN3, ADC1/2_INP/INN8, ADC1/2_INP/INN2, ADC1/2_INN/INP2, ADC1/2_INP/INN0, ADC1/2_INP/INN2, ADC1/2_INP/INN2, ADC1/2_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to *Section 26.4.17: Stopping an ongoing conversion (ADSTP, JADSTP)*).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC_CFGR register). Refer to *Section 26.4.21: Queue of context for injected conversions*

26.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

T_{CONV} = Sampling time + 12.5 ADC clock cycles

Example:

With $F_{adc_ker_ck}$ = 30 MHz and a sampling time of 2.5 ADC clock cycles:

T_{CONV} = (2.5 + 12.5) ADC clock cycles = 15 ADC clock cycles = 500 ns

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Note: Depending on the ADC conversion mode, the real sampling time can vary compared to the SMP value programmed above, while the equivalent total conversion time (T_{CONV}) does not change:

- *For the first conversion in scan or continuous mode and all the conversions in discontinuous mode, the real sampling time is 0.5 clock cycle less compared to the value configured above.*
- *For the second and subsequent conversions in scan or continuous mode, 0.5 cycle is added to the configured sampling time. This additional 0.5 clock cycle overlaps with the previous conversion cycle.*

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

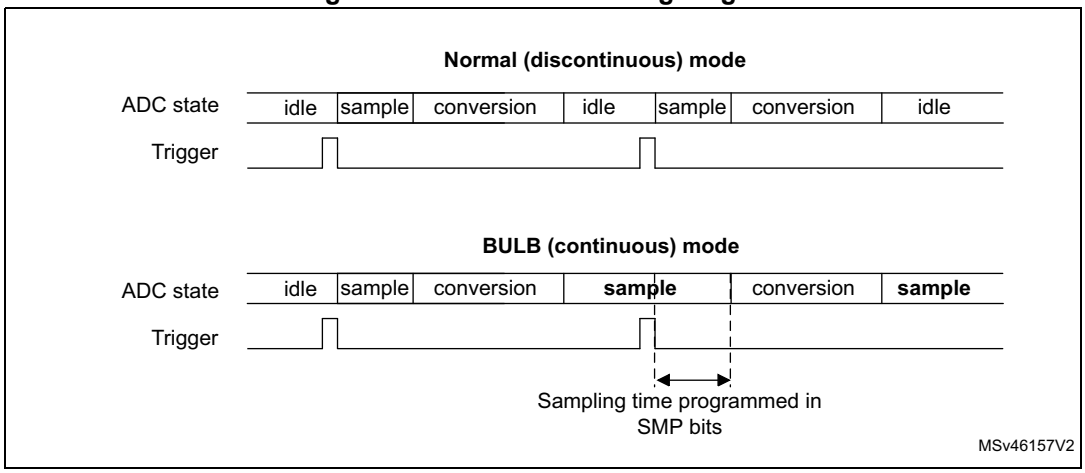
When the BULB bit is set in ADC register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion, after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

The bulb mode is neither compatible with the continuous conversion mode nor with the injected channel conversion.

When the BULB bit is set, it is not allowed to set SMPTRIG bit in ADC_CFGR2.

Figure 201. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed though SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected, each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

When a software trigger is selected, the software trigger is not the ADSTART bit in ADC_CR but the SWTRIG bit. SWTRIG bit has to be set to start the sampling period, and the SWTRIG bit has to be cleared to end the sampling period and start the conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is neither compatible with the continuous conversion mode, nor with the injected channel conversion.

When SMPTRIG bit is set, it is not allowed to set BULB bit.

I/O analog switch voltage booster

The resistance of the I/O analog switches increases when the V_{DDA} voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low V_{DDA}

by enabling an internal voltage booster through BOOSTE bit or by selecting a V_{DD} booster voltage (if V_{DD} > 2.7 V) through the ADV_READY bit of the PWR_PMCR register.

SMPPLUS control bit

The SMPPLUS bit can be used to change the sampling time from 2.5 ADC clock cycles to 3.5 ADC clock cycles.

26.4.13 Single conversion mode (CONT = 0)

In single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:

- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 16-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

Note: To convert a single channel, program a sequence with a length of 1.

26.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section).

26.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN is not equal to 0x0

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN is not equal to 0x0

Note: In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT = x, EXTSEL = x)
  - After execution of the ADSTP procedure asserted by the software.

Note: In continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTSEL # 0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is cleared by hardware:

- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- in all cases (JEXTSEL = x)
  - After execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.

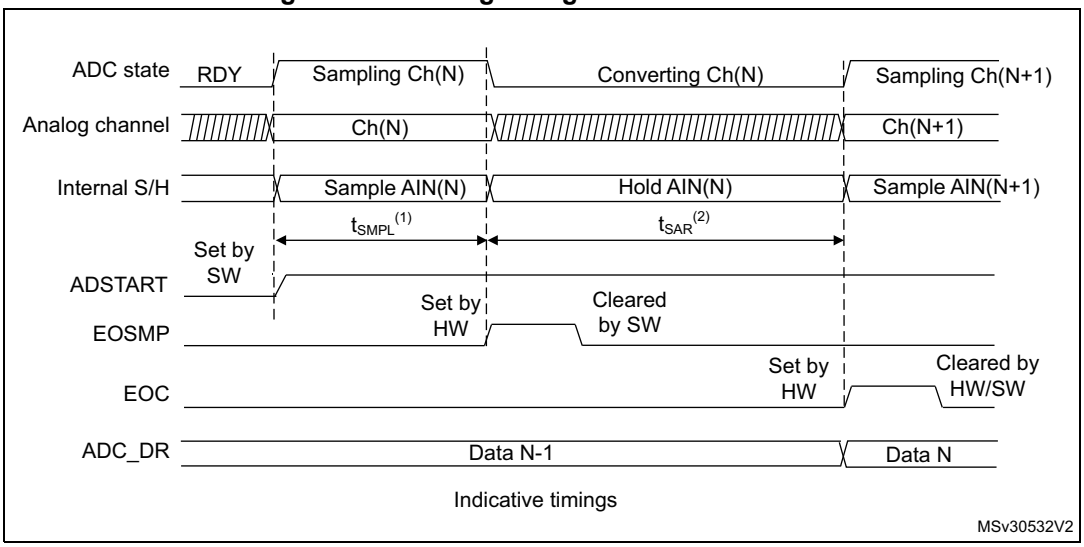
26.4.16 ADC timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [2.5\ _{|min} + 12.5\ _{|12bit}] \times T_{ADC_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 83.33\ ns\ _{|min} + 416.67\ ns\ _{|12bit} = 500.0\ ns$ (for $F_{ADC_CLK} = 30$ MHz)

Figure 202. Analog-to-digital conversion time



1. T_{SMPL} depends on SMP[2:0].
2. T_{SAR} depends on RES[2:0].

26.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 203. Stopping ongoing regular conversions

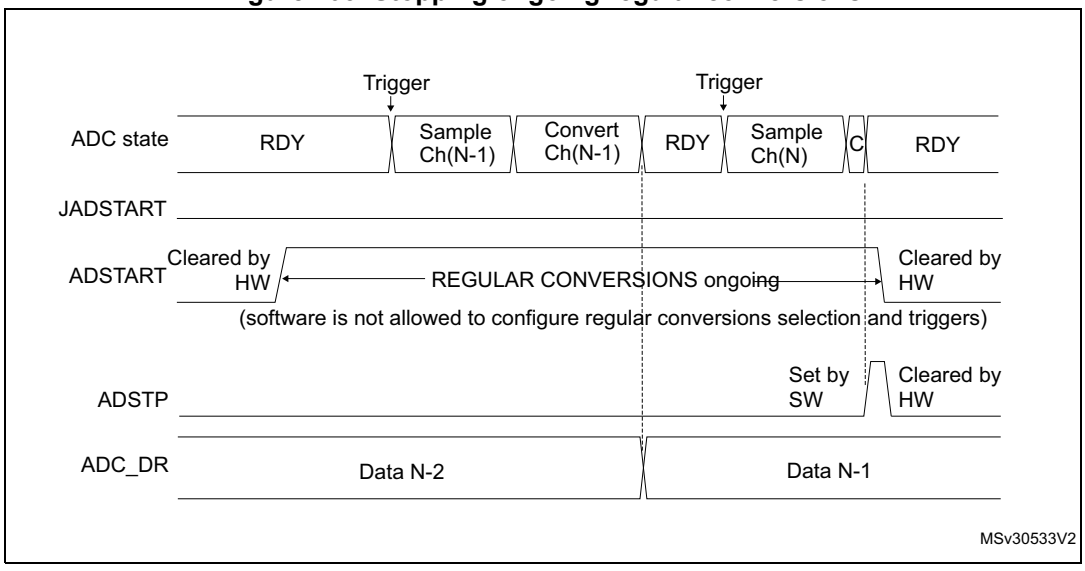
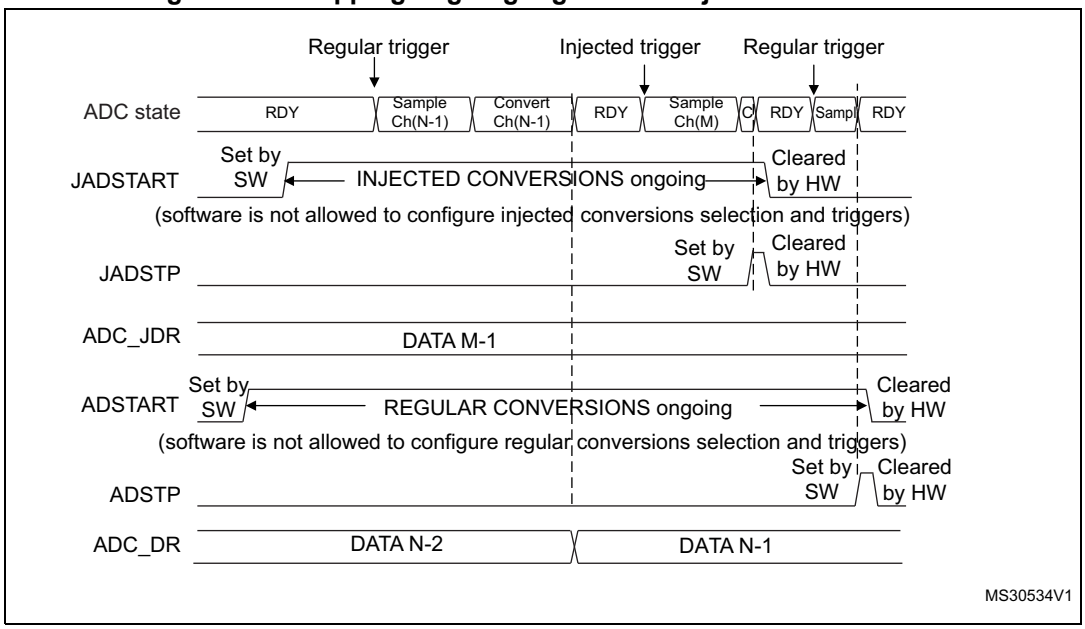


Figure 204. Stopping ongoing regular and injected conversions



26.4.18 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (such as timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.

- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

Table 258 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 258. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

Table 259. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	– If JQDIS = 1 (Queue disabled): Hardware trigger detection disabled, software trigger detection enabled – If JQDIS = 0 (Queue enabled), Hardware and software trigger detection disabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

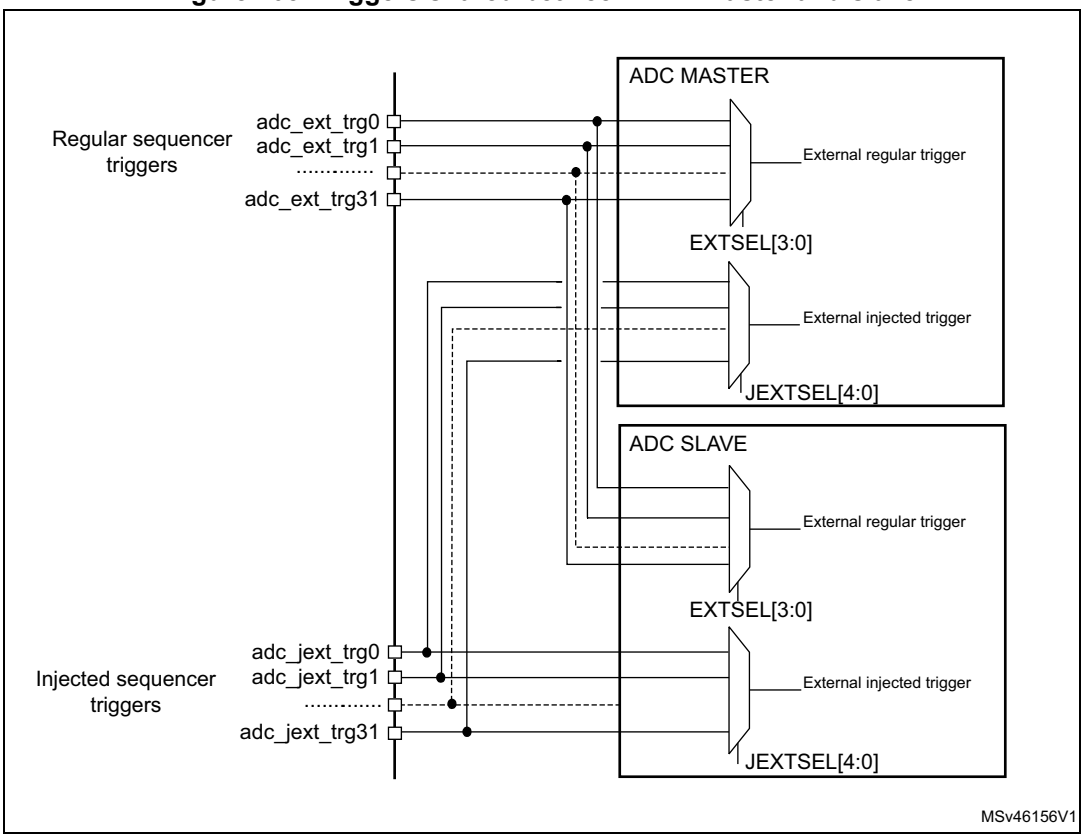
Note: The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to Section 26.4.21: Queue of context for injected conversions.

The EXTSEL and JEXTSEL control bits select which out of 32 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly. The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 26.4.21: Queue of context for injected conversions on page 1057.

Figure 205. Triggers shared between ADC master and slave



Refer to Table ADC interconnection in *Section 26.4.2: ADC pins and internal signals* for the list of all the external triggers that can be used for regular conversion.

26.4.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 206* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

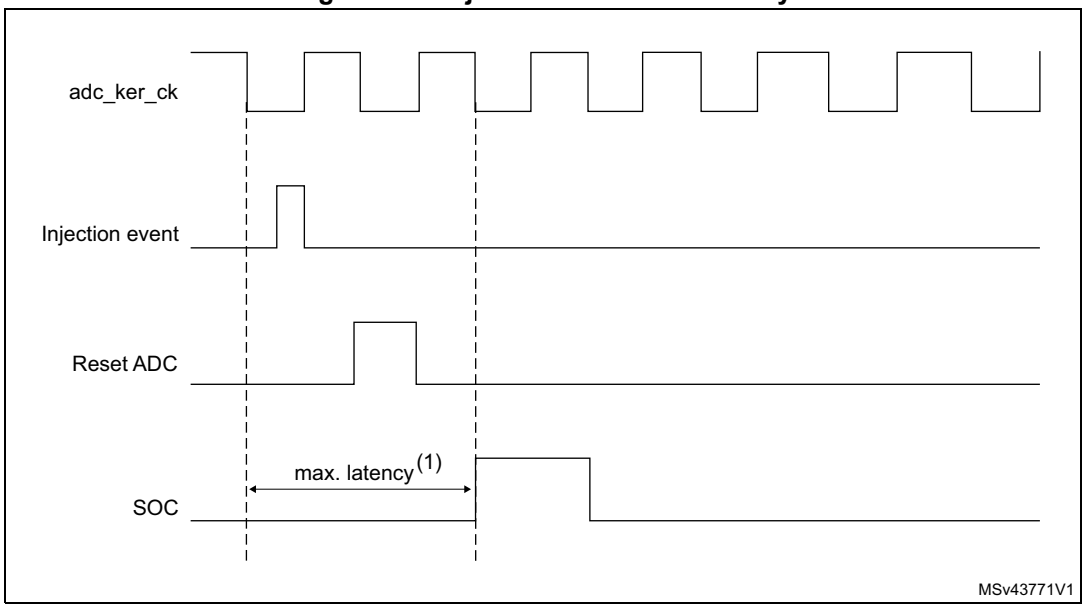
In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer's data in JAUTO mode, it is necessary to program it in circular mode. If the single-shot mode is selected, the JAUTO sequence is stopped upon DMA Transfer Complete event.

Figure 206. Injected conversion latency



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

26.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

26.4.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:

- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)

All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The ADC_JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the ADC_JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the ADC_JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next ADC_JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register ADC_JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of ADC_JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM = 0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  - If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into ADC_JSQR register.
- Reading ADC_JSQR register returns the current ADC_JSQR context which is active at that moment. When the ADC_JSQR context is empty, JSQi is read as 0x00.
- The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling the ADC by setting ADDIS = 1:
  - If JQM = 0, the Queue is maintained with the last active context.
  - If JQM = 1, the Queue becomes empty and triggers are ignored.

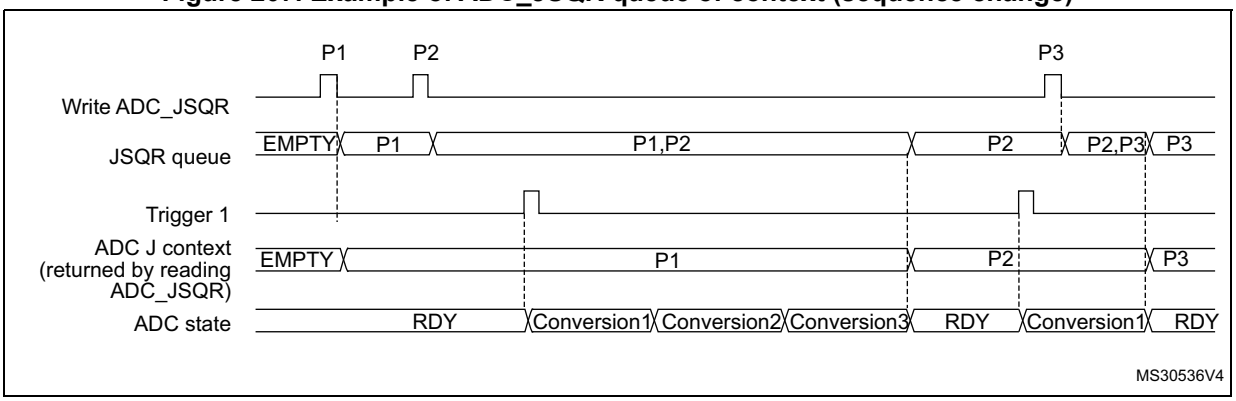
Note: When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue.The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- *1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out*
- *2nd trigger, disc. Sequence 1: 2nd conversion.*
- *3rd trigger, discontinuous. Sequence 1: 3rd conversion.*
- *4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.*
- *5th trigger, discontinuous. Sequence 2: 2nd conversion.*
- *6th trigger, discontinuous. Sequence 2: 3rd conversion.*

Behavior when changing the trigger or sequence context

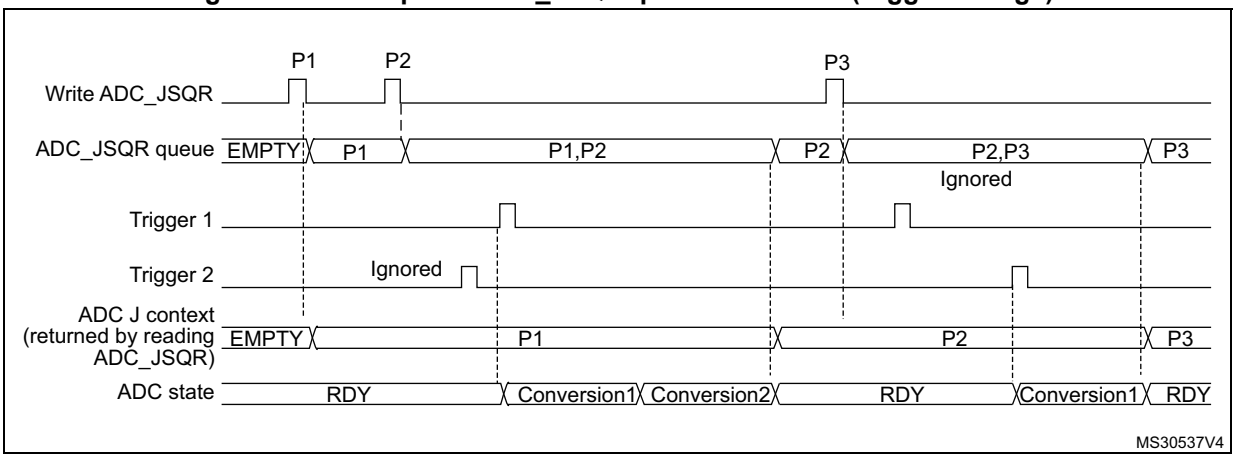
Figure 207 and *Figure 208* show the behavior of the context Queue when changing the sequence or the triggers.

Figure 207. Example of ADC_JSQR queue of context (sequence change)



1. Parameters:
   P1: sequence of 3 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 4 conversions, hardware trigger 1

Figure 208. Example of ADC_JSQR queue of context (trigger change)

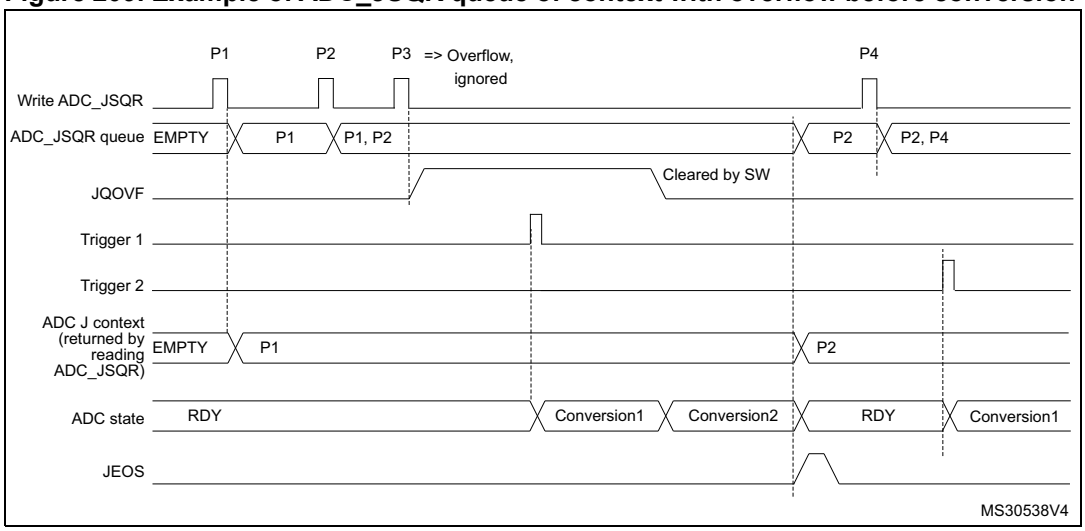


1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 4 conversions, hardware trigger 1

Queue of context: Behavior when a queue overflow occurs

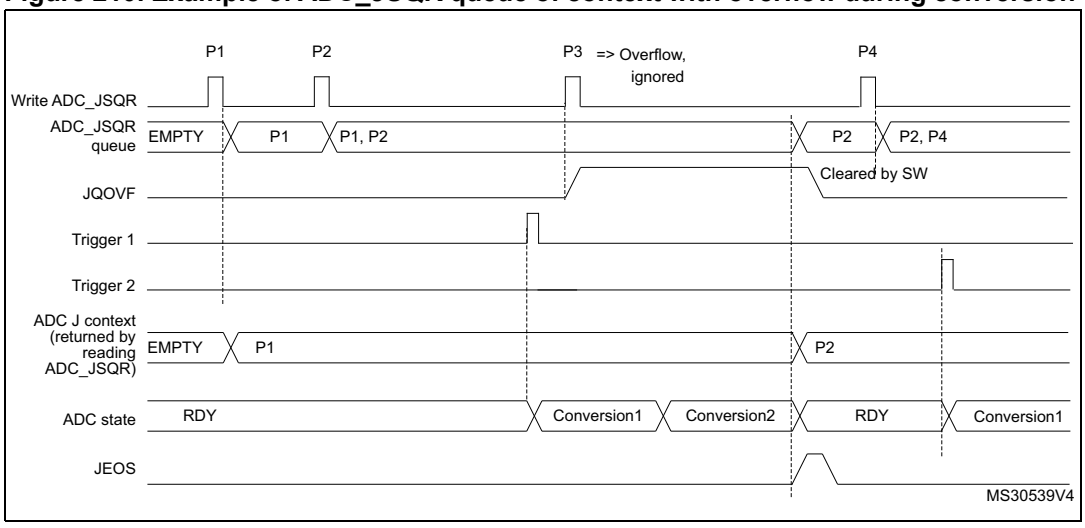
The *Figure 209* and *Figure 210* show the behavior of the context Queue if an overflow occurs before or during a conversion.

Figure 209. Example of ADC_JSQR queue of context with overflow before conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

Figure 210. Example of ADC_JSQR queue of context with overflow during conversion



1. Parameters:
   P1: sequence of 2 conversions, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 2
   P3: sequence of 3 conversions, hardware trigger 1
   P4: sequence of 4 conversions, hardware trigger 1

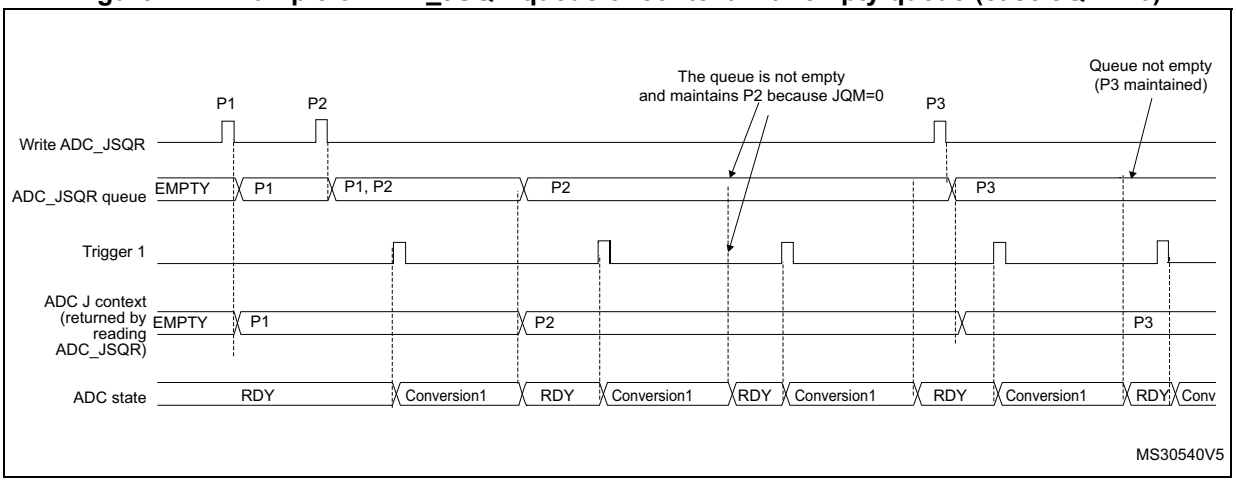
It is recommended to manage the queue overflows as described below:

- After each P context write into ADC_JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

Queue of context: Behavior when the queue becomes empty

Figure 211 and *Figure 212* show the behavior of the context Queue when the Queue becomes empty in both cases JQM = 0 or 1.

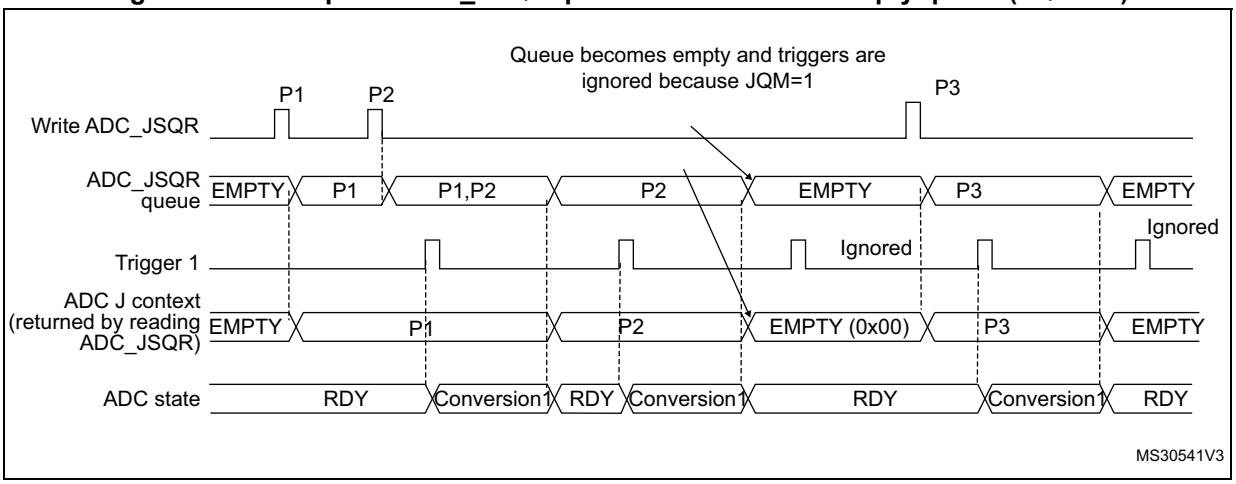
Figure 211. Example of ADC_JSQR queue of context with empty queue (case JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Note: When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.

Figure 212. Example of ADC_JSQR queue of context with empty queue (JQM = 1)

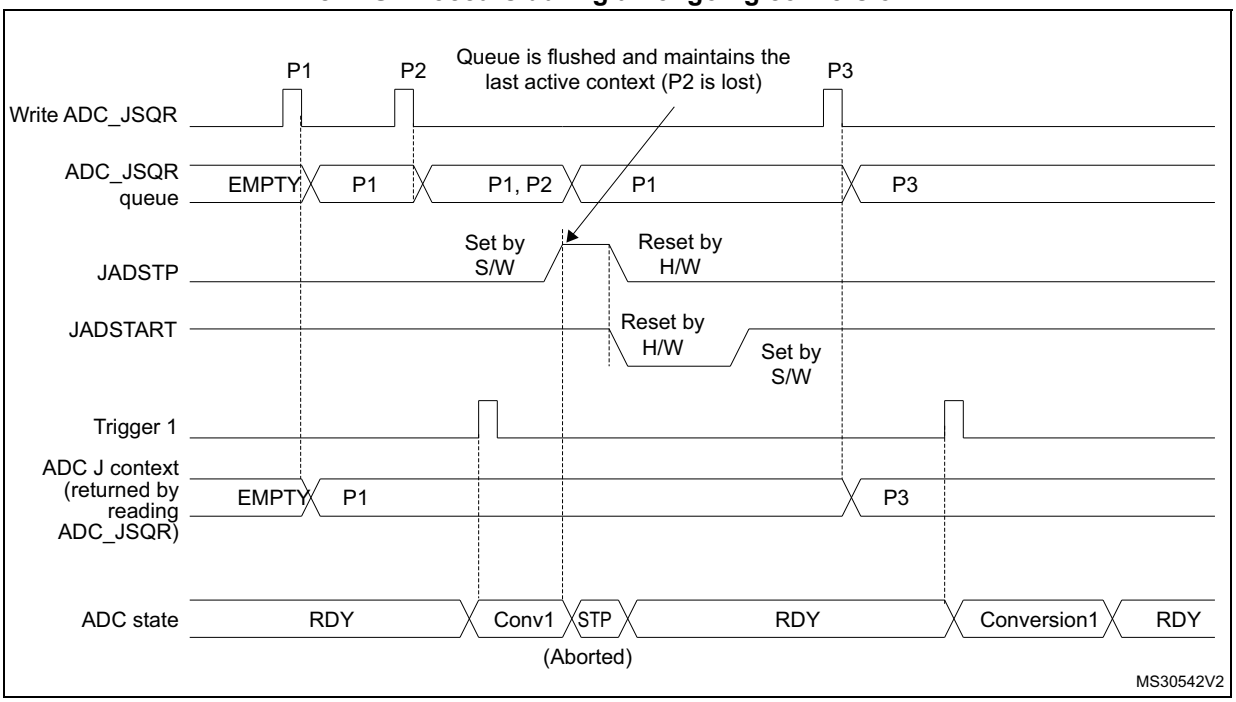


1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Flushing the queue of context

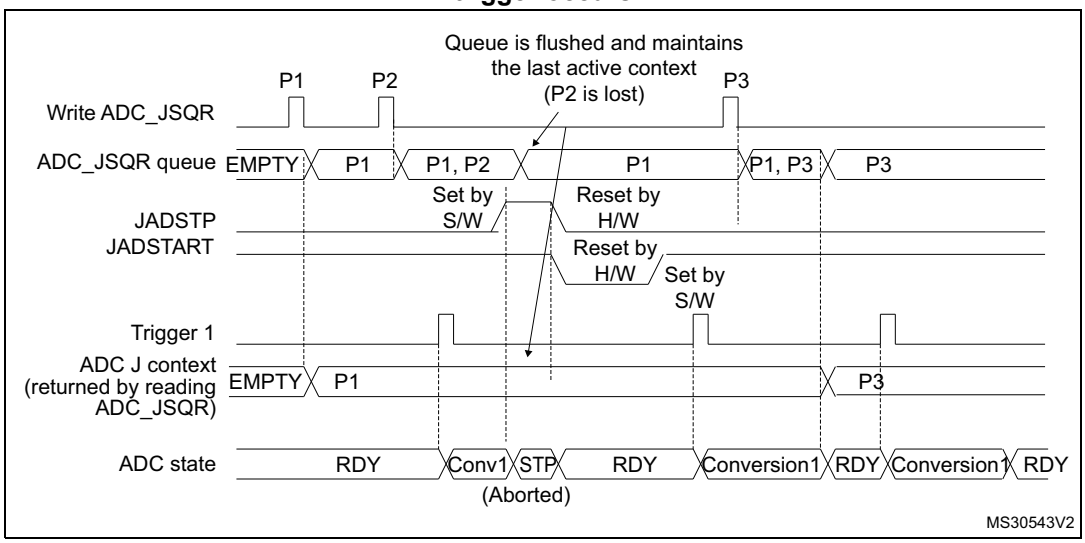
The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 213. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.



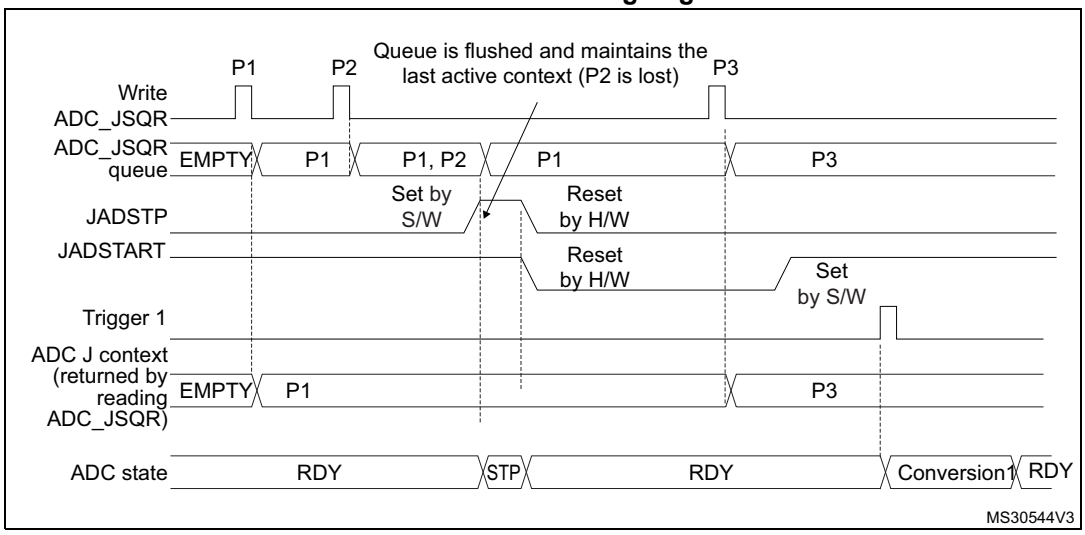
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 214. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs



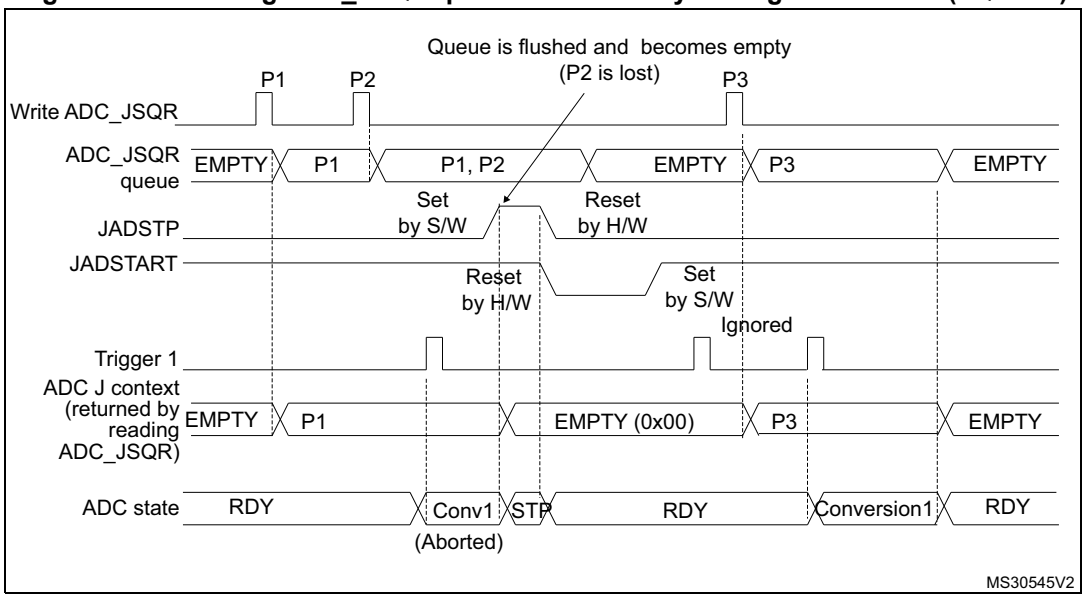
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 215. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion



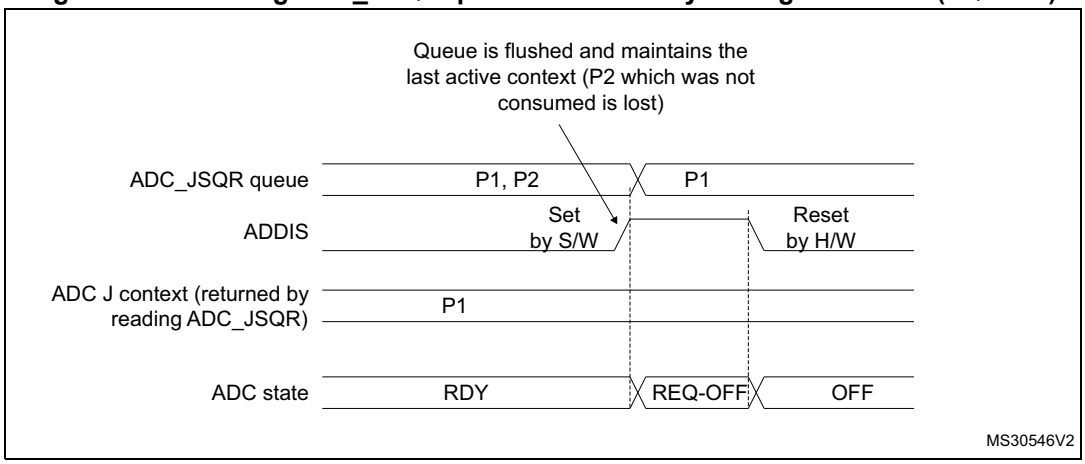
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 216. Flushing ADC_JSQR queue of context by setting JADSTP = 1 (JQM = 1)



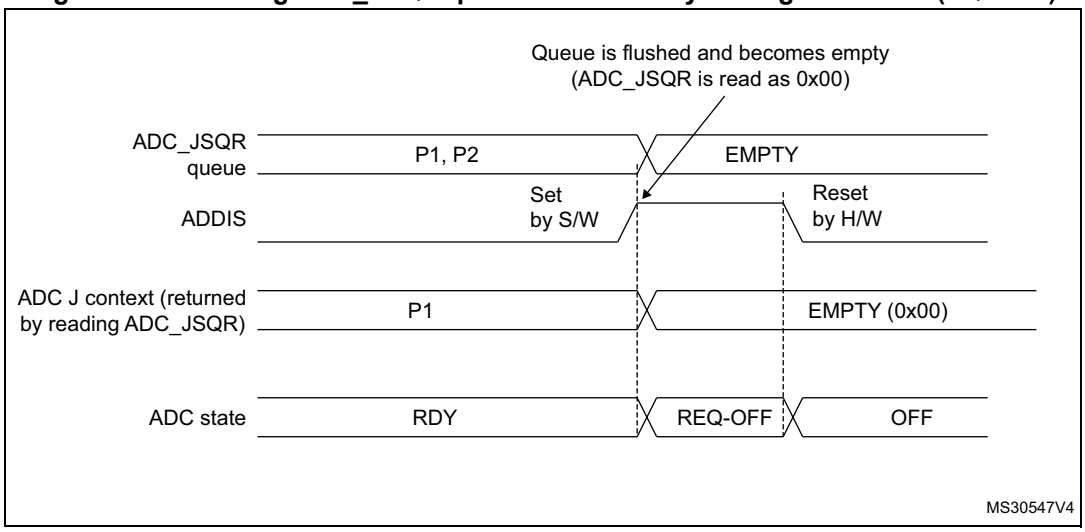
1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 217. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 0)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Figure 218. Flushing ADC_JSQR queue of context by setting ADDIS = 1 (JQM = 1)



1. Parameters:
   P1: sequence of 1 conversion, hardware trigger 1
   P2: sequence of 1 conversion, hardware trigger 1
   P3: sequence of 1 conversion, hardware trigger 1

Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy ADC_JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC_CFGR register.

26.4.22 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. *Figure 223*, *Figure 224*, *Figure 225* and *Figure 226* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 260*.

Table 260. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at F_{ADC} = 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling Time = 2.5 ADC clock cycles)	T_{CONV} (ns) at F_{ADC} = 30 MHz
12	12.5 ADC clock cycles	416.67 ns	15 ADC clock cycles	500.0 ns
10	10.5 ADC clock cycles	350.0 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	11 ADC clock cycles	366.67 ns
6	6.5 ADC clock cycles	216.67 ns	9 ADC clock cycles	300.0 ns

26.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

26.4.24 End of conversion sequence (EOS, JEOS)

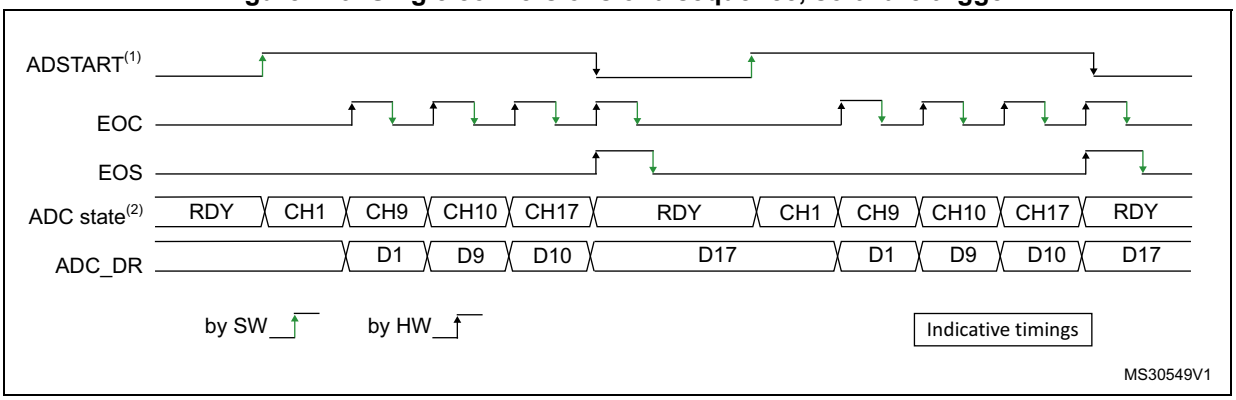
The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

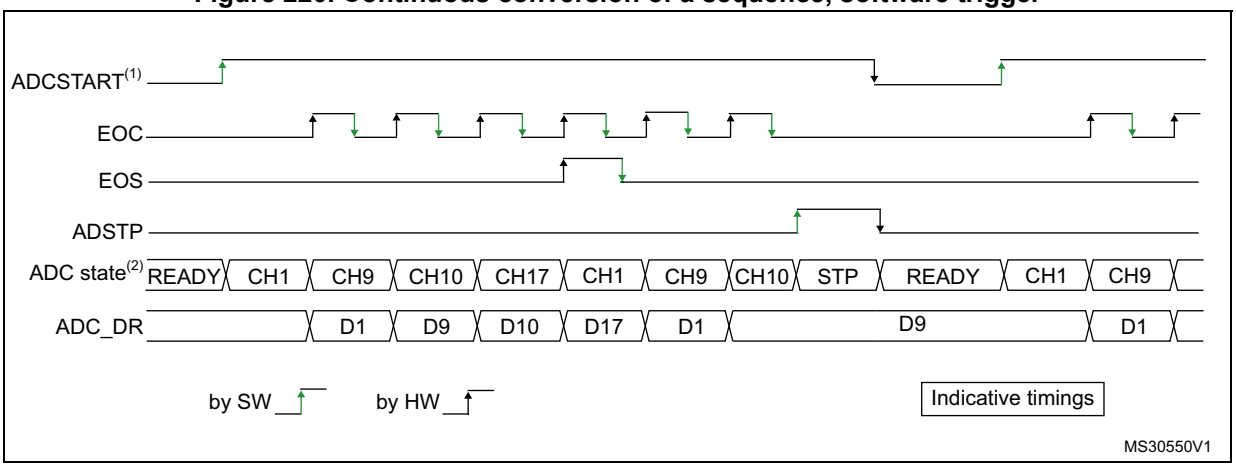
26.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

Figure 219. Single conversions of a sequence, software trigger



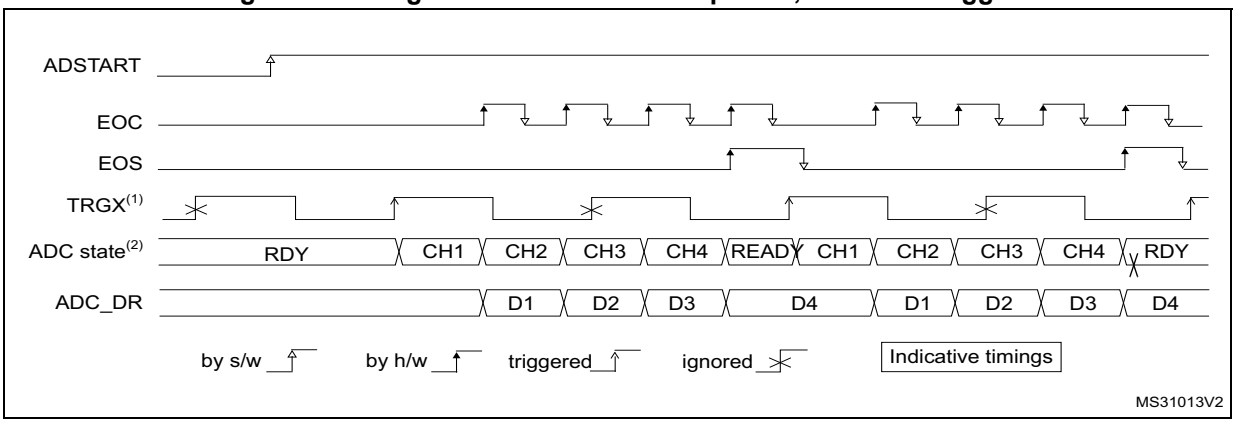
1. EXTEN = 0x0, CONT = 0
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 220. Continuous conversion of a sequence, software trigger



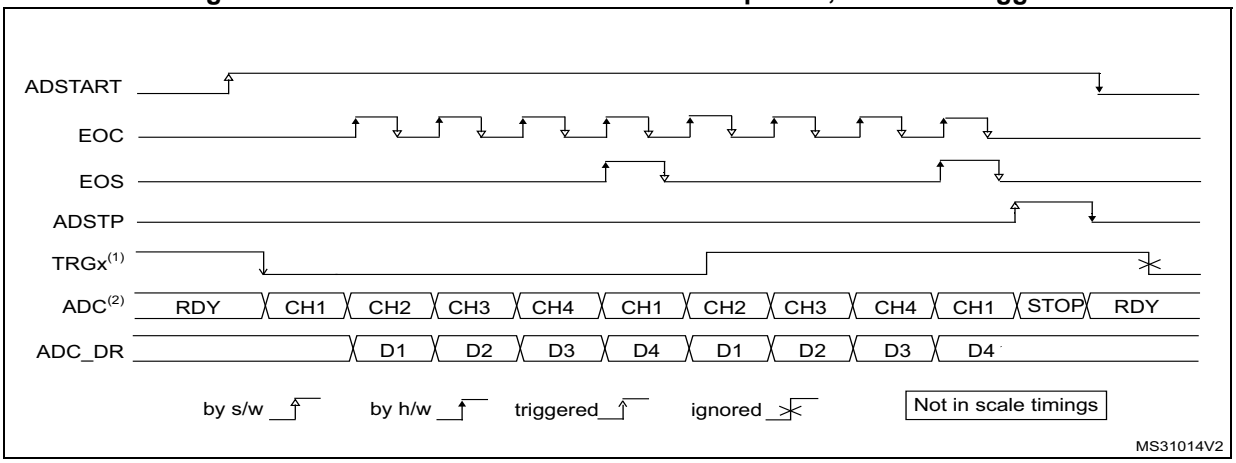
1. EXTEN = 0x0, CONT = 1
2. Channels selected = 1,9, 10, 17; AUTDLY = 0.

Figure 221. Single conversions of a sequence, hardware trigger



1. TRGx (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 222. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

26.4.26 Data management

Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in *Figure 223*, *Figure 224*, *Figure 225* and *Figure 226*.

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in *Figure 225* and *Figure 226*.

Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.

Offset

An offset y (y = 1,2,3,4) can be applied to a channel by setting the bit OFFSET_EN = 1 into ADC_OFRy register. The channel to which the offset is to be applied is programmed into the bits OFFSET_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSET[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).

Table 263 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 261. Offset computation versus data resolution

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed 12-bit data	-
01: 10-bit	DATA[11:2],00	OFFSET[11:0]	Signed 10-bit data	The user must configure OFFSET[1:0] to “00”

Table 261. Offset computation versus data resolution (continued)

Resolution (bits RES[1:0])	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
10: 8-bit	DATA[11:4],0000	OFFSET[11:0]	Signed 8-bit data	The user must configure OFFSET[3:0] to “0000”
11: 6-bit	DATA[11:6],000000	OFFSET[11:0]	Signed 6-bit data	The user must configure OFFSET[5:0] to “000000”

When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y = 1,2,3,4) corresponding to the channel “i”:

- If one of the offsets is enabled (bit OFFSET_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 223, *Figure 224*, *Figure 225* and *Figure 226* show alignments for signed and unsigned data.

Figure 223. Right alignment (offset disabled, unsigned value)

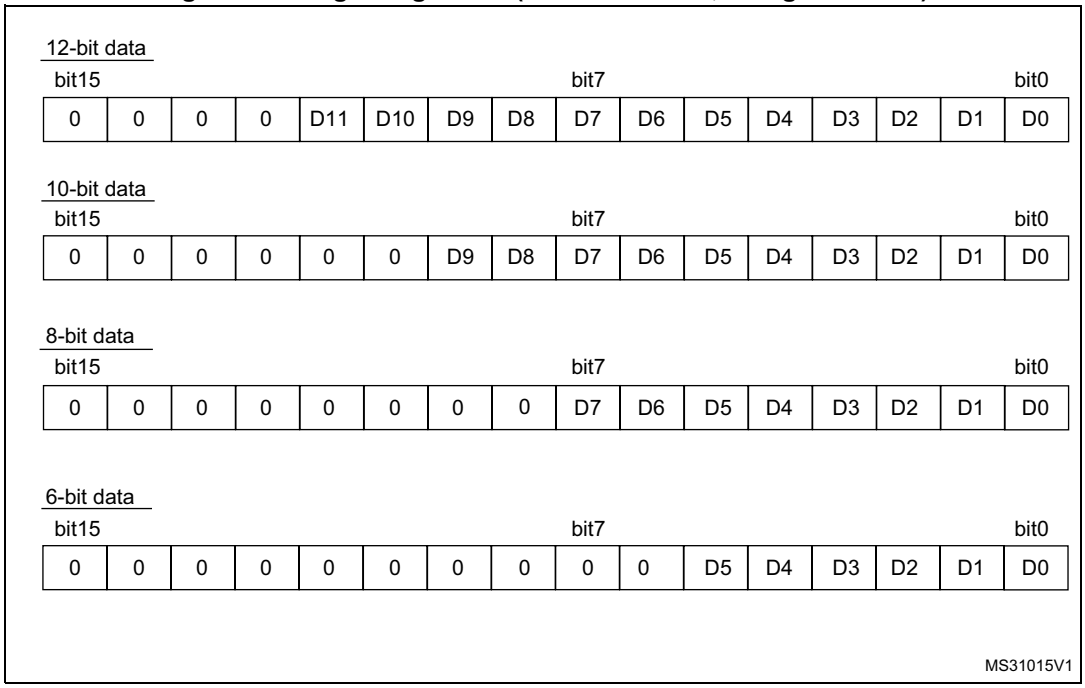


Figure 224. Right alignment (offset enabled, signed value)

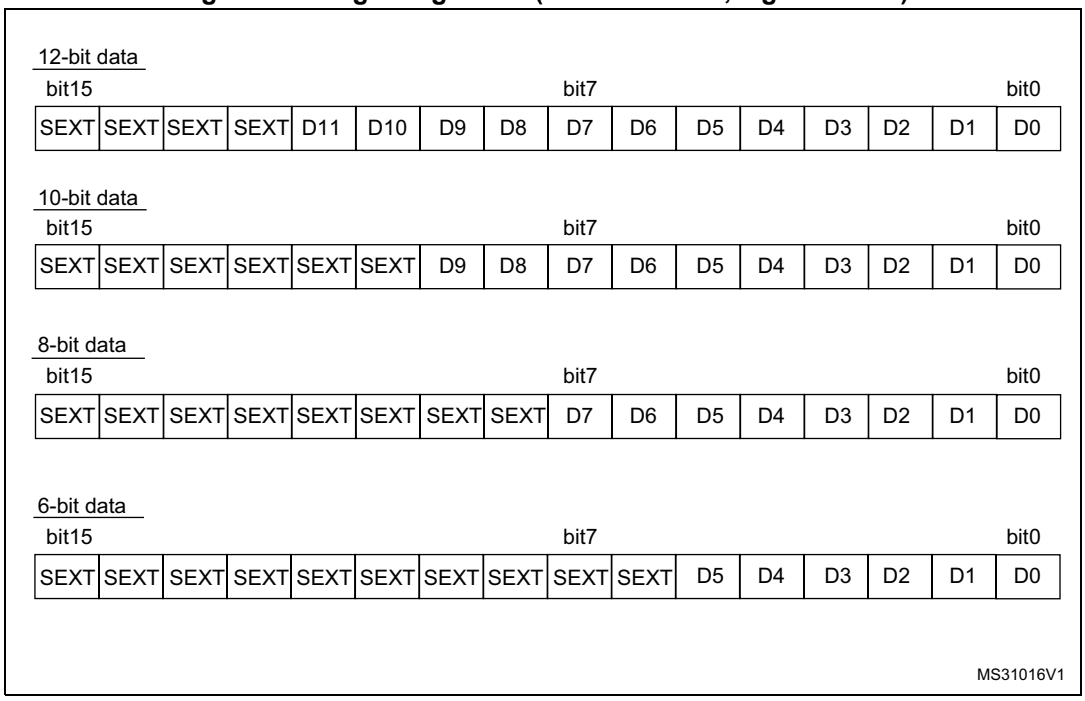


Figure 225. Left alignment (offset disabled, unsigned value)

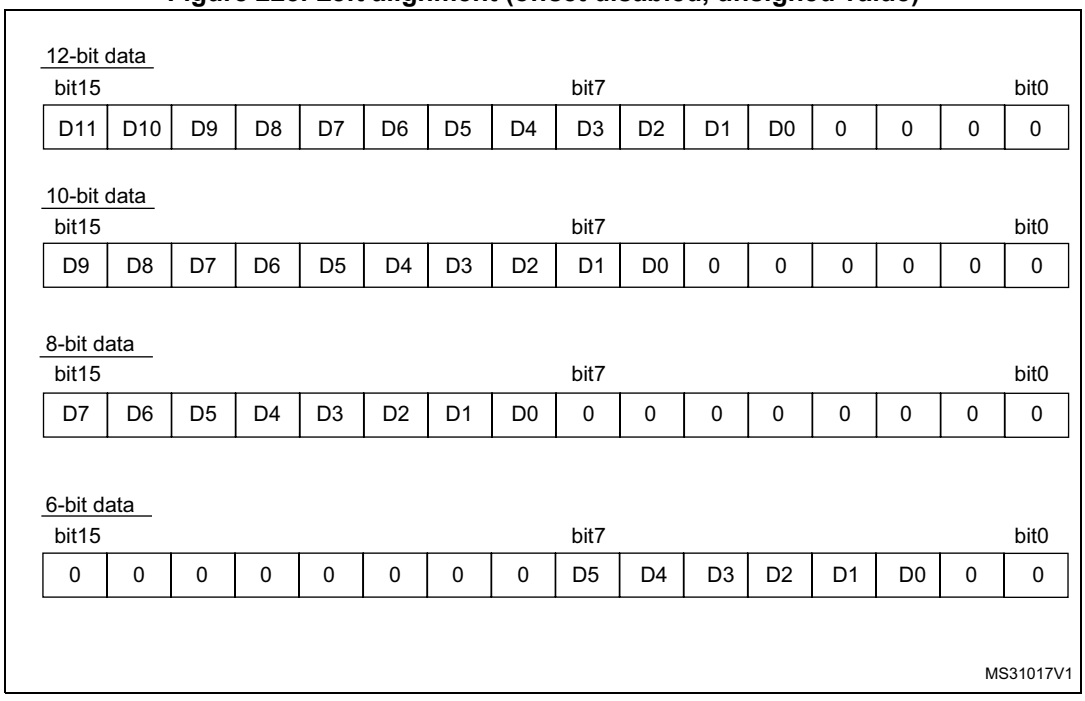
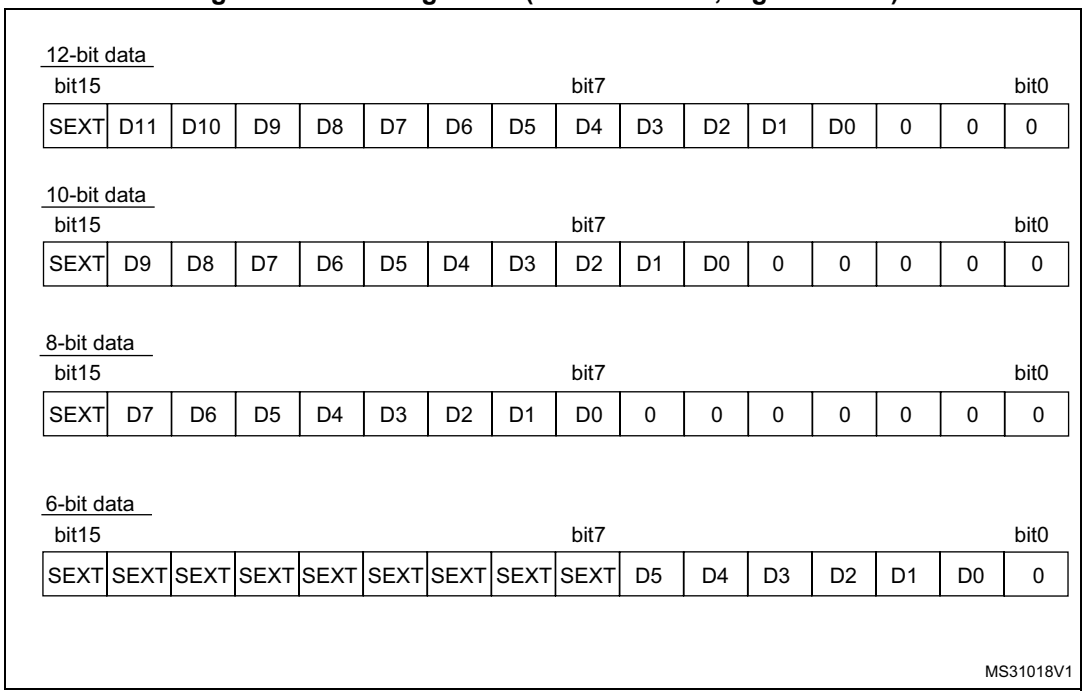


Figure 226. Left alignment (offset enabled, signed value)



Offset compensation

When SATEN bit is set in ADC_OFRy register during offset operation, data are unsigned. All the offset data saturate at 0x000 (in 12-bit mode). When OFFSETPOS bit is set, the offset direction is positive and the data saturate at 0xFFF (in 12-bit mode). In 8-bit mode, data saturate at 0x00 and 0xFF, respectively.

The analog watchdog comparison is performed before the offset compensation.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (three stages) is overflowed.

The OVR flag is set when a new conversion completes while ADC_CR register FIFO was full. An interrupt is generated if OVRIE bit is set to 1.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP to 1. Since ADC_DR FIFO features three stages, up to three data are stored in the FIFO.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (three stages) and the new conversion is discarded and lost. In this mode, ADC_DR FIFO is disabled. If the FIFO is full, any further conversion is performed but the resulting data is also discarded. EOC

is cleared by reading ADC_DR register. However, the FIFO can still contain previously converted data.

- OVRMOD = 1: The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversions is performed normally and the ADC_DR register always contains the latest converted data.

Figure 227. Example of overrun (OVRMOD = 0)

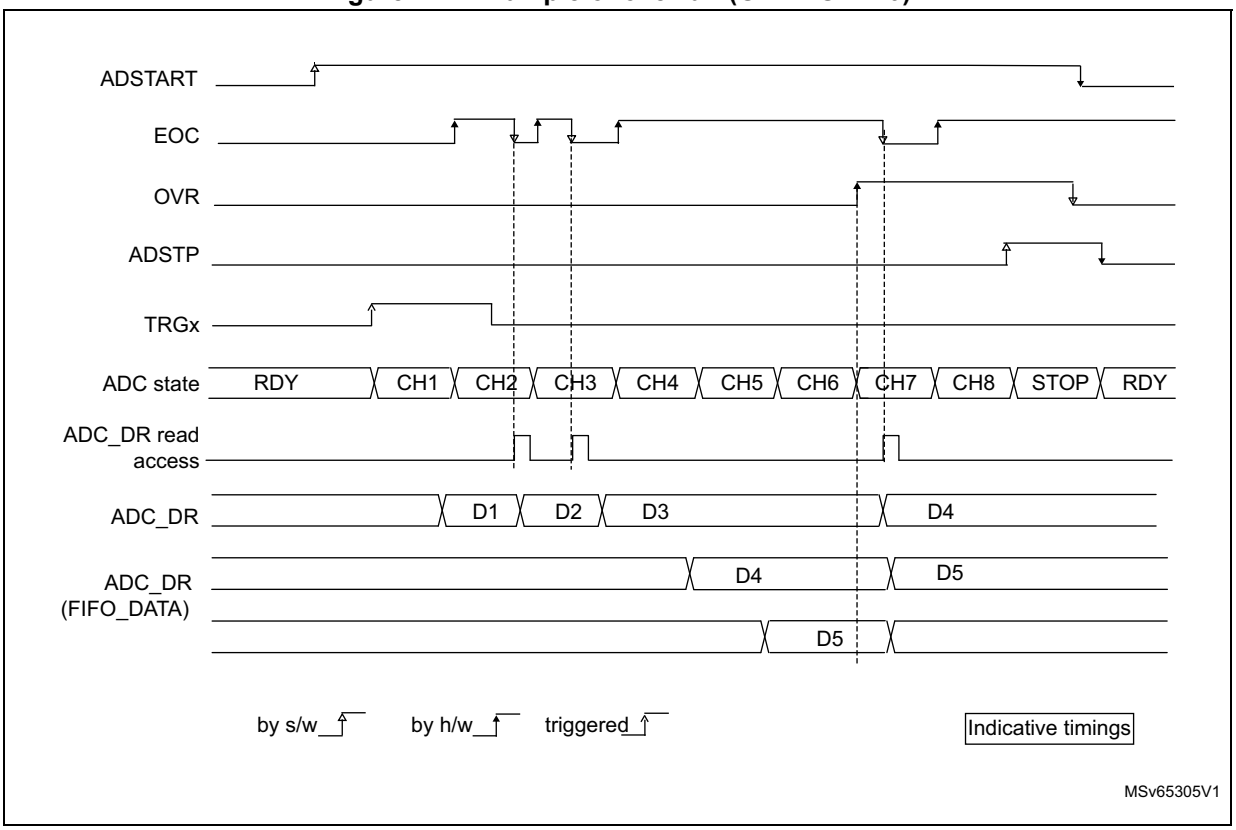
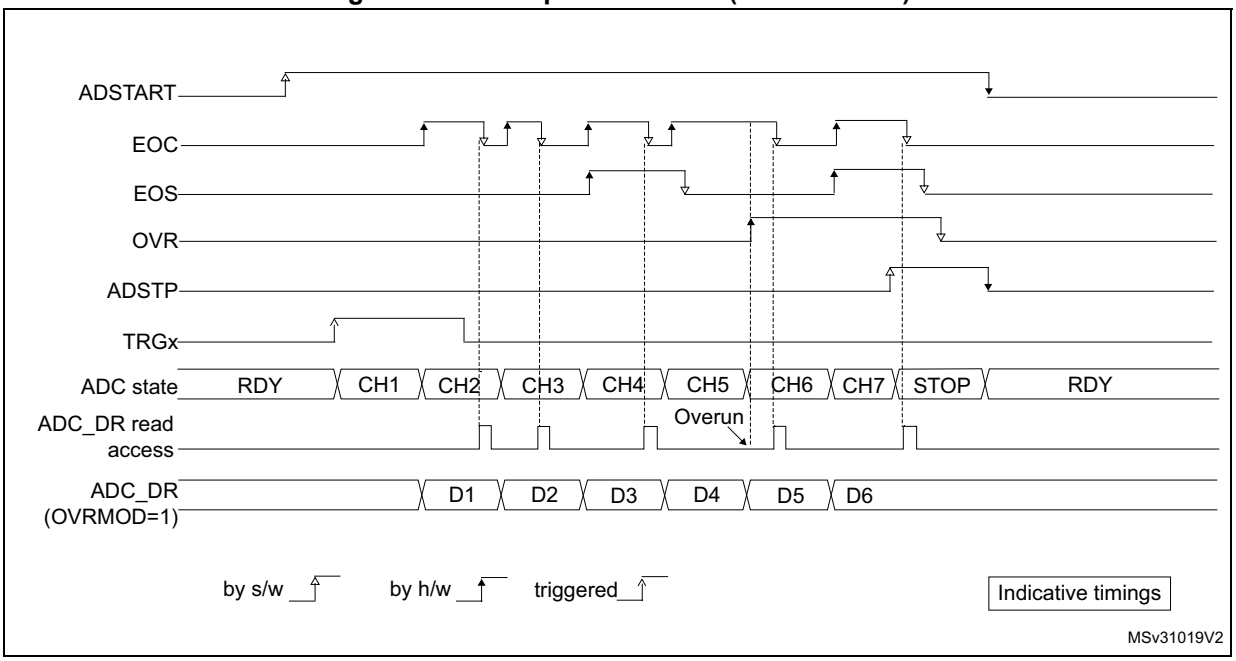


Figure 228. Example of overrun (OVRMOD = 1)



Note: There is no overrun detection on the injected channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.

Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG of the ADC_CFGR register in single ADC mode, or with bit DMACFG of the ADC_CCR register in dual ADC mode:

- DMA one shot mode (DMACFG = 0).
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG = 1)
  This mode is suitable when programming the DMA in circular mode.

DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

26.4.27 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY = 1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see *Figure 229*).
- For an injected conversion: when the JEOS bit has been cleared (see *Figure 230*).

This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see *Figure 230*).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see *Figure 232*).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 233*).

To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

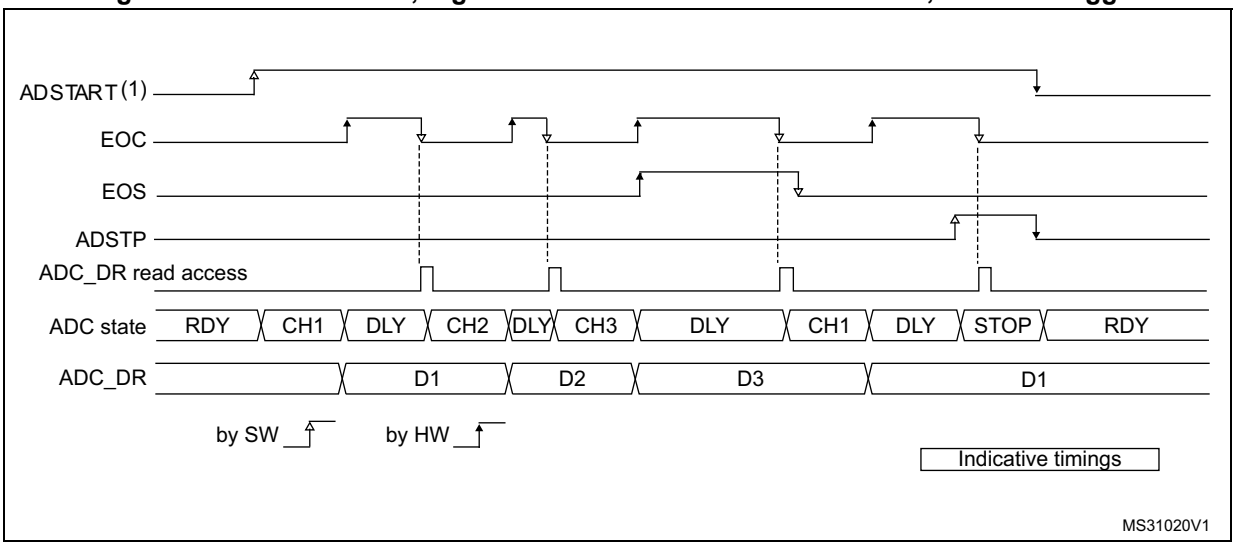
1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP = 1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

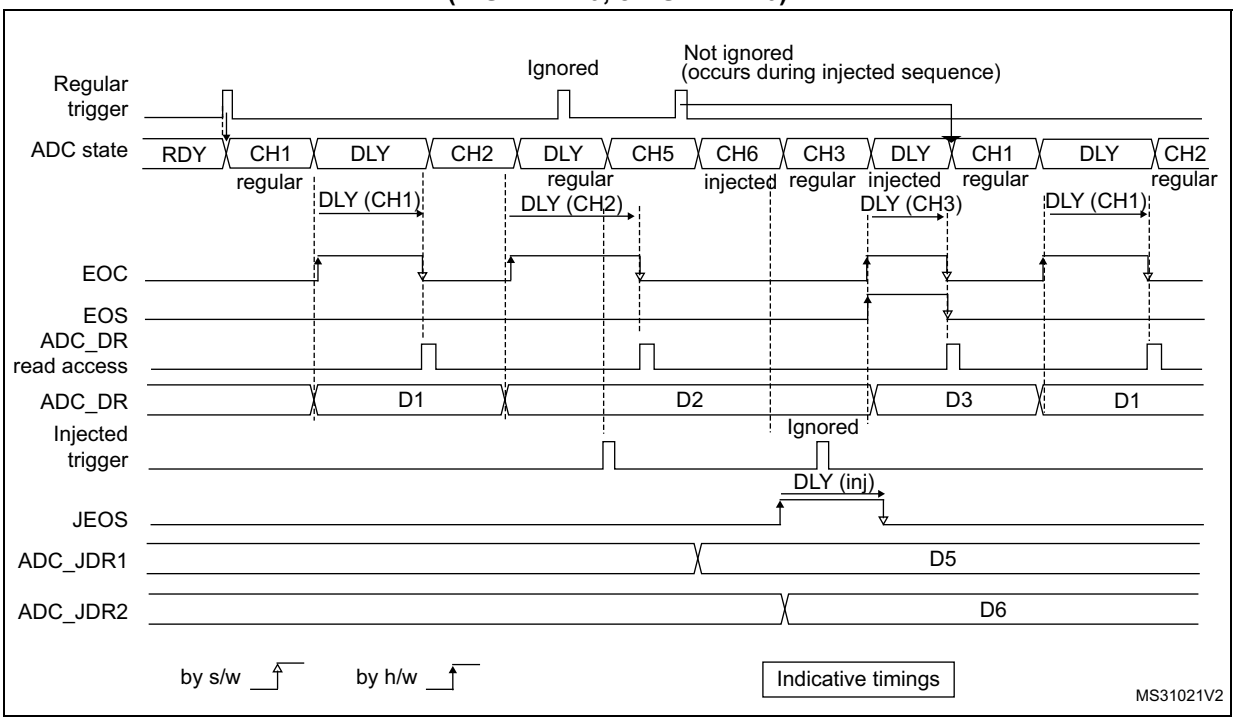
In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

Figure 229. AUTODLY = 1, regular conversion in continuous mode, software trigger



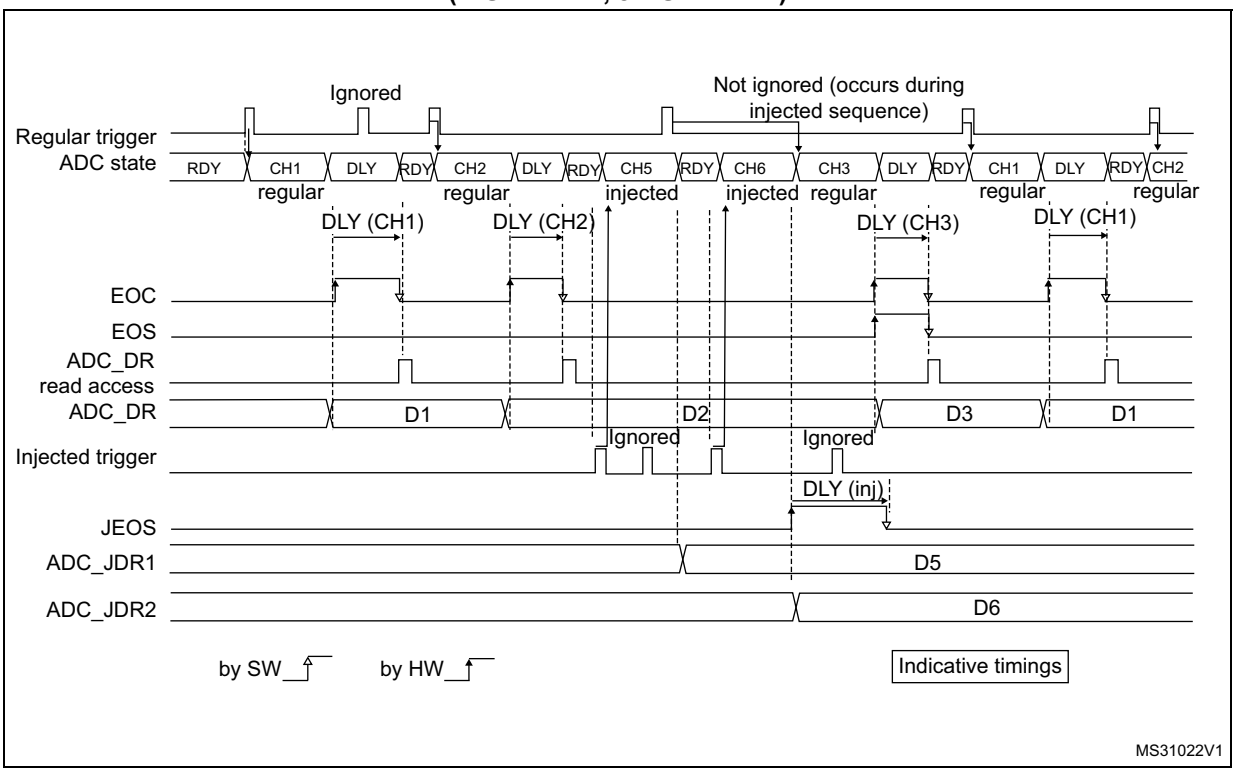
1. AUTDLY = 1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT = 1, CHANNELS = 1,2,3
3. Injected configuration DISABLED

Figure 230. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



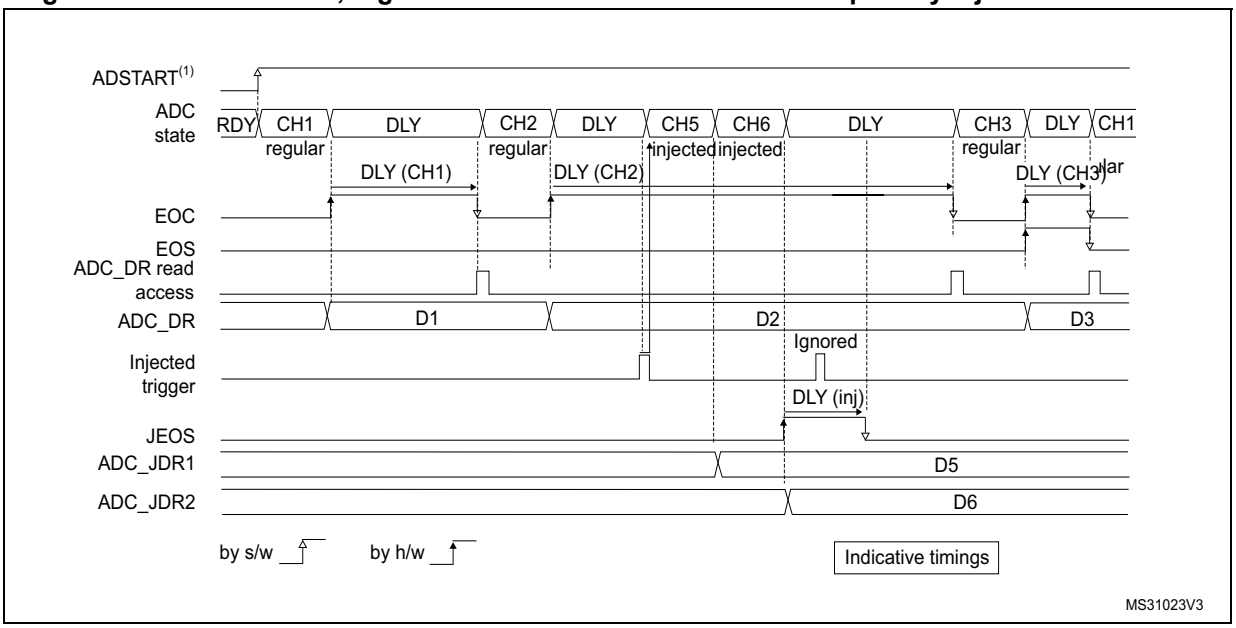
1. AUTDLY = 1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

Figure 231. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



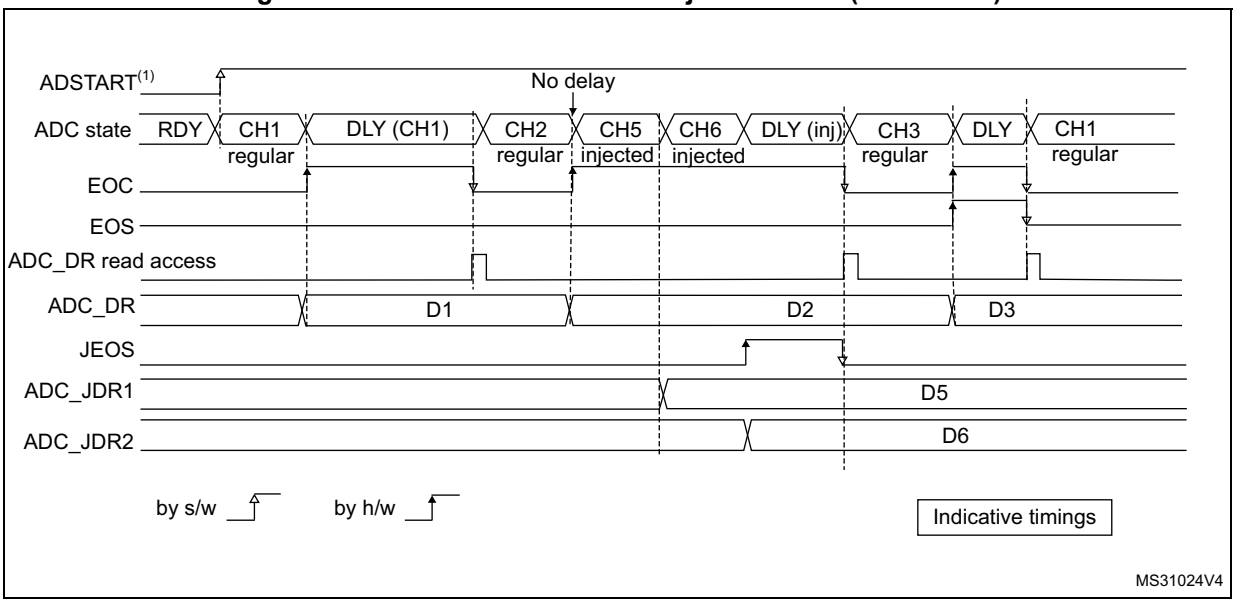
1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x1 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 1, CHANNELS = 5,6

Figure 232. AUTODLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5,6

Figure 233. AUTODLY = 1 in auto- injected mode (JAUTO = 1)

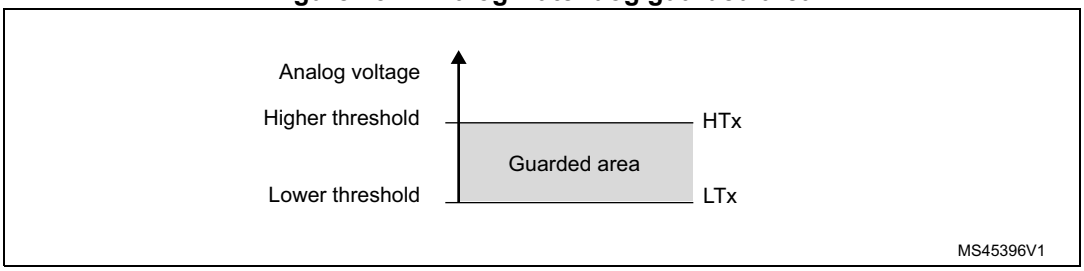


1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2
3. Injected configuration: JAUTO = 1, CHANNELS = 5,6

26.4.28 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 234. Analog watchdog guarded area



AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels(1) remain within a configured voltage range (window).

Table 262 shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

Table 262. Analog watchdog channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned) before the offset compensation stage.

Table 263 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

Table 263. Analog watchdog 1 comparison

Resolution(bit RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:0]	LT1[11:0] and HT1[11:0]	-
01: 10-bit	DATA[11:2],00	LT1[11:0] and HT1[11:0]	User must configure LT1[1:0] and HT1[1:0] to 00
10: 8-bit	DATA[11:4],0000	LT1[11:0] and HT1[11:0]	User must configure LT1[3:0] and HT1[3:0] to 0000
11: 6-bit	DATA[11:6],000000	LT1[11:0] and HT1[11:0]	User must configure LT1[5:0] and HT1[5:0] to 000000

Analog watchdog filter for watchdog 1

When an ADC is configured with only one input channel (selecting several channels in scan mode not allowed), a valid ADC conversion data filter can be configured:

- When the converted data belong to the interval defined in ADC_TR1, the AWD1 flag remains cleared.
- If data are out-of-range a number of times higher than the value specified in AWDFILT bit of ADC_TR1, the AWD1 flag is set an the corresponding interrupt is issued.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDxCH[19:0] (x = 2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[19:0] (x = 2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. *Table 264* describes how the comparison is performed for all the possible resolutions.

Table 264. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
00: 12-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:0] are not relevant for the comparison
01: 10-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	DATA[3:2] are not relevant for the comparison

Table 264. Analog watchdog 2 and 3 comparison

Resolution (bits RES[1:0])	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned	Thresholds	
10: 8-bit	DATA[11:4]	LTx[7:0] and HTx[7:0]	-
11: 6-bit	DATA[11:6],00	LTx[7:0] and HTx[7:0]	User must configure LTx[1:0] and HTx[1:0] to 00

ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y = ADC number, x = watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:

- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS = 1). Note that stopping regular or injected conversions (setting ADSTP = 1 or JADSTP = 1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 235. ADCy_AWDx_OUT signal generation (on all regular channels)

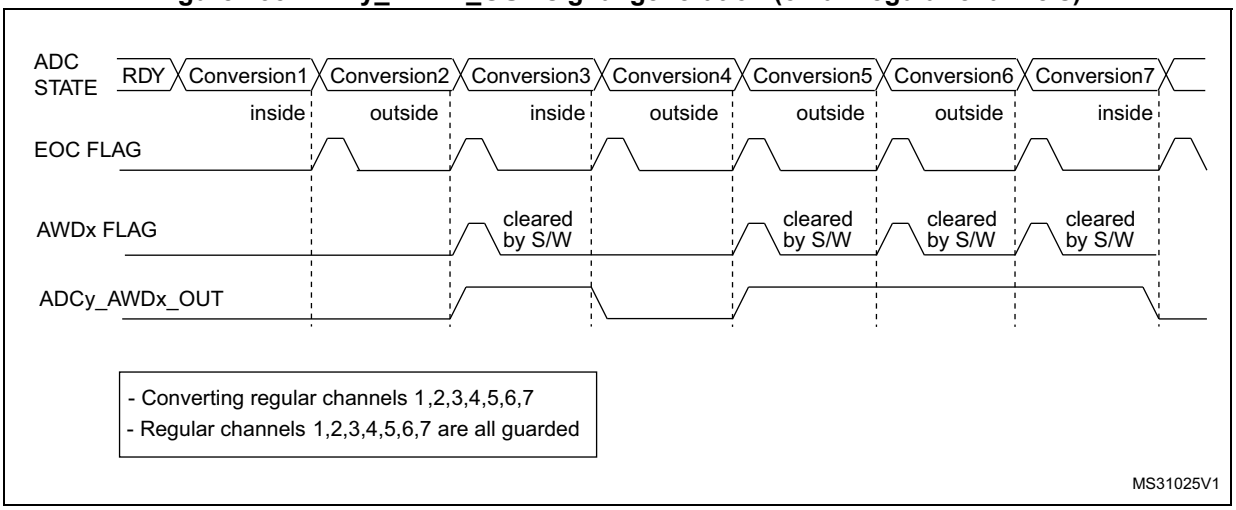


Figure 236. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software)

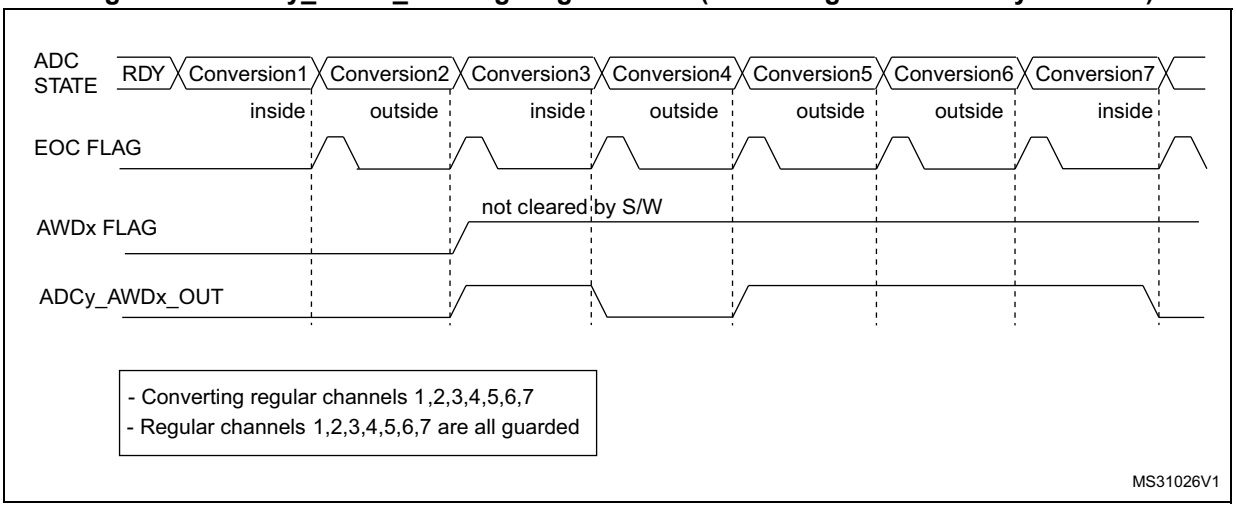


Figure 237. ADCy_AWDx_OUT signal generation (on a single regular channel)

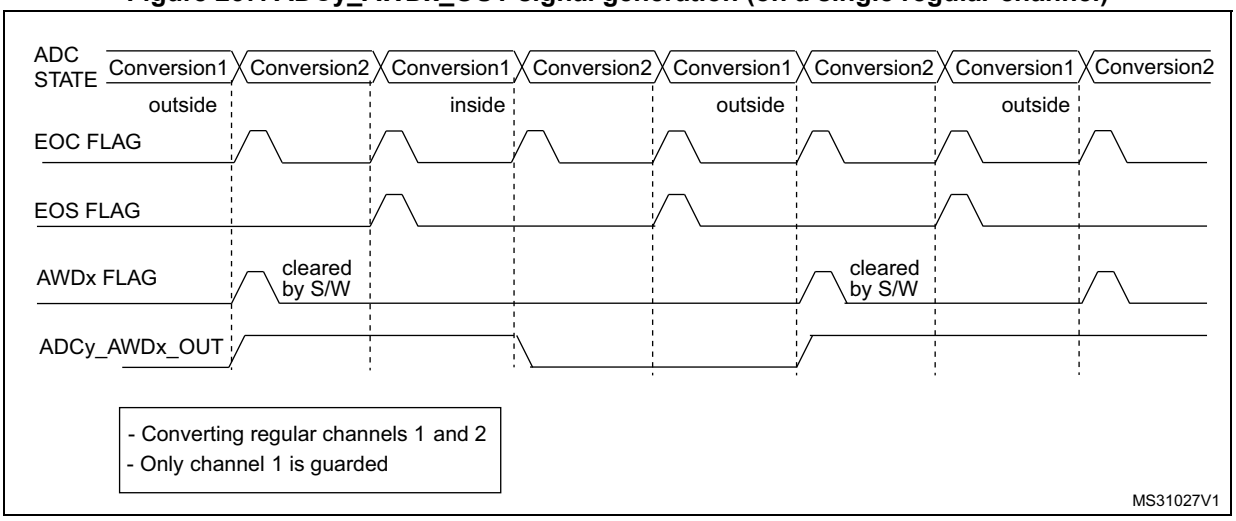
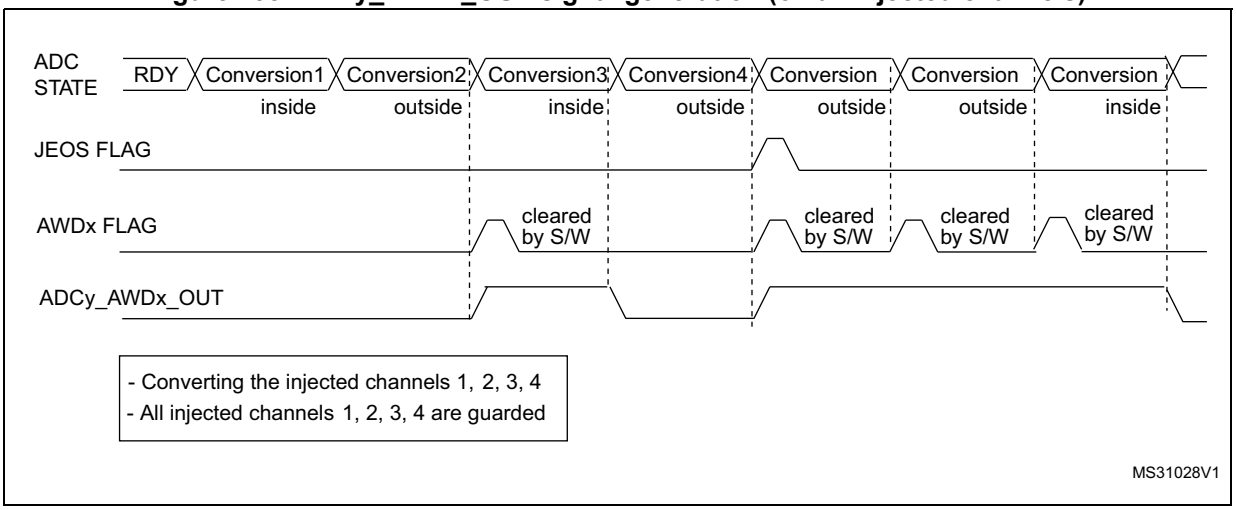


Figure 238. ADCy_AWDx_OUT signal generation (on all injected channels)



Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTx[11:0] and HTx[11:0] are updated during the ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking is removed at the next start of conversion, resulting in a analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new interval, no interrupt and AWDx_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion which started after the threshold update. If AWD_xOUT is already asserted, programming the new thresholds does not deassert the AWDx_OUT signal.

To update both LTx[11:0] and HTx[11:0] during an ADC conversion, a unique write access to the ADC_TRx register be performed.

Analog watchdog with offset compensation

When the offset compensation is enabled, the analog watchdog compares the threshold before the data compensation.

26.4.29 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to the 16 least significant bits, rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

Note: If the intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.

Figure 239. 20-bit to 16-bit result truncation

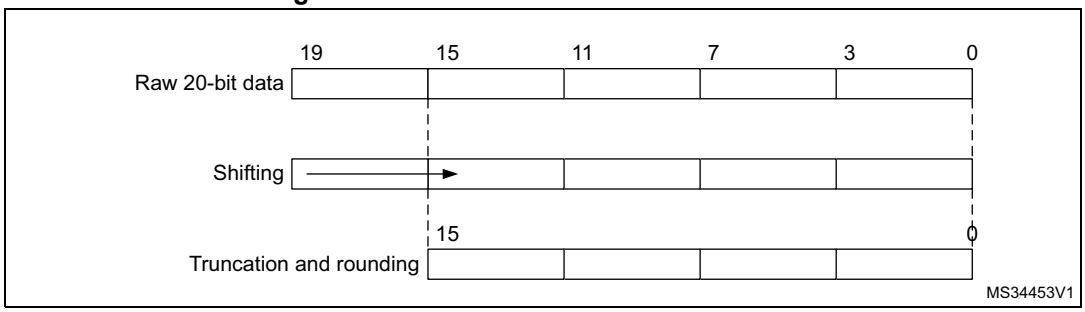


Figure 240 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 240. Numerical example with 5-bit shift and rounding

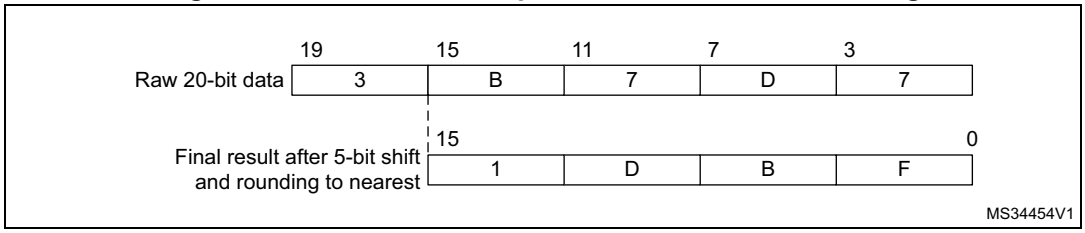


Table 265 gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 265. Maximum output results versus N and M (gray cells indicate truncation)

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS = 0111	8-bit shift OVSS = 1000
2x	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x020
4x	0x3FFC	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
8x	0x7FF8	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080
16x	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
32x	0x1FFE0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200
64x	0x3FFC0	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
128x	0x7FF80	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800
256x	0xFFF00	0xFF00	0xFF80	0xFFC0	0xFFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N

conversions, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follow:

- The end of the sampling phase (EOSMP) is set after each sampling phase
- The end of conversion (EOC) occurs once every N conversions, when the oversampled result is available
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after N x sequence length conversions total)

ADC operating modes supported when oversampling (single ADC mode)

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous conversion modes
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

Note: The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR is ignored and the data are always provided right-aligned.

Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).

Analog watchdog

The analog watchdog functionality is maintained, with the following difference:

- The RES[1:0] bits are ignored, the comparison is always done by using the full 12-bit values HT1[11:0] and LT1[11:0] for AWD1, and the 8-MSB value of HT2/HT3[7:0] and LT2/LT3[7:0] for AWD2 and AWD3.
- The comparison is done on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4] for AWD1, and ADC_DR[15:8] for AWD2 and ADW3

Note: Care must be taken when using high shifting values, this reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[7:0] (AWD1/2/3), with HT1[11:8] and LT1[11:8] kept reset (AWD1 only).

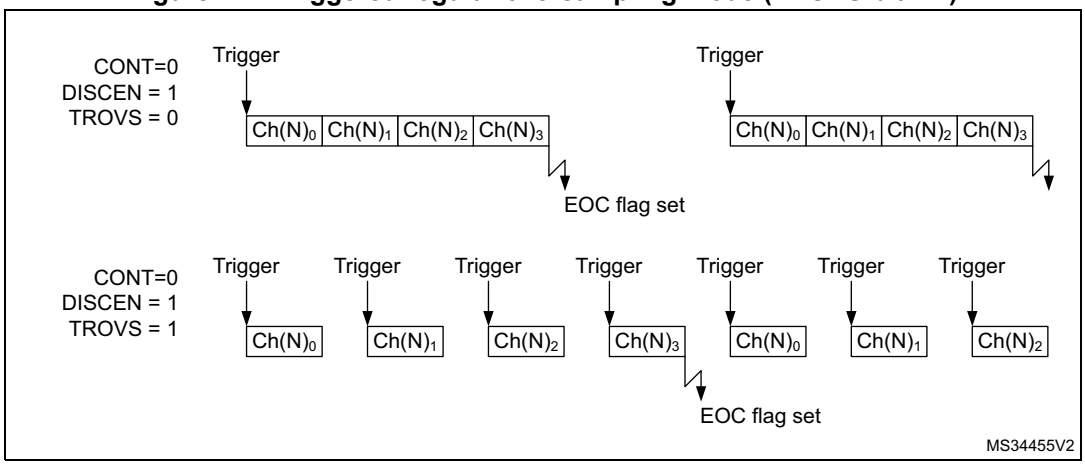
Triggered mode

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The *Figure 241* below shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 241. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

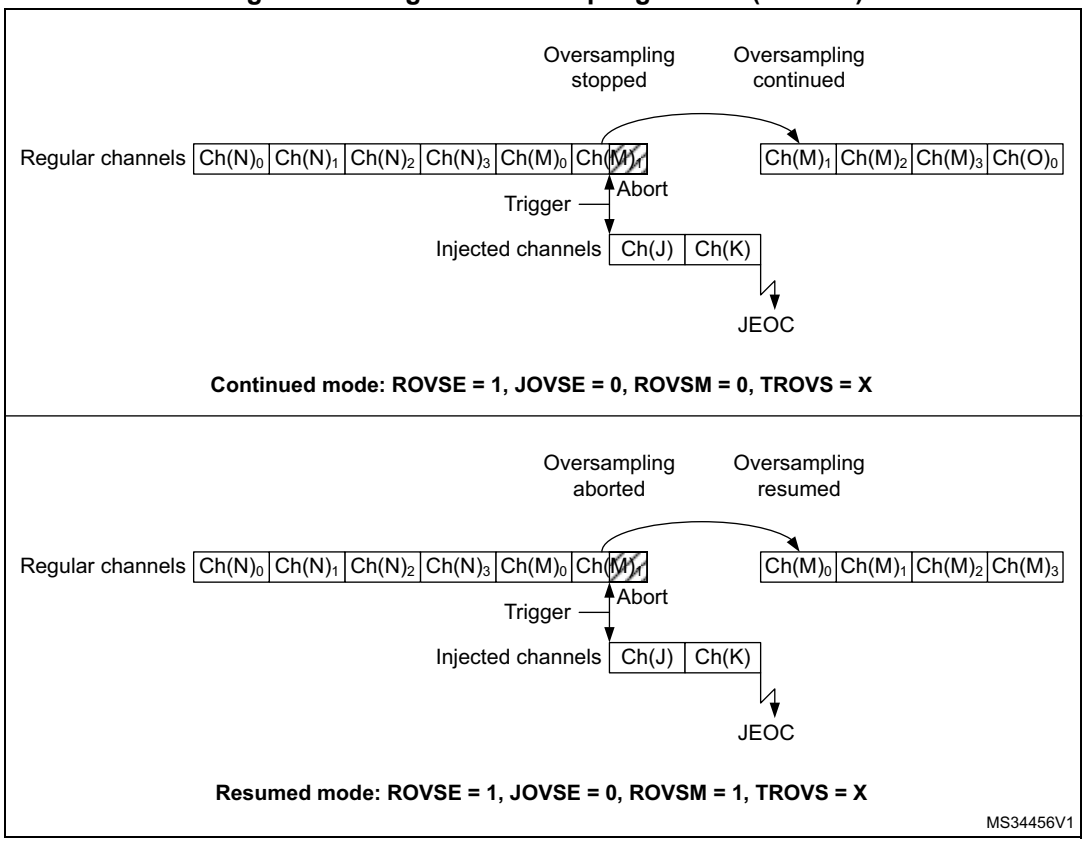
Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be completed and the regular sequencer is blocked.

The *Figure 242* gives examples for a 4x oversampling ratio.

Figure 242. Regular oversampling modes (4x ratio)



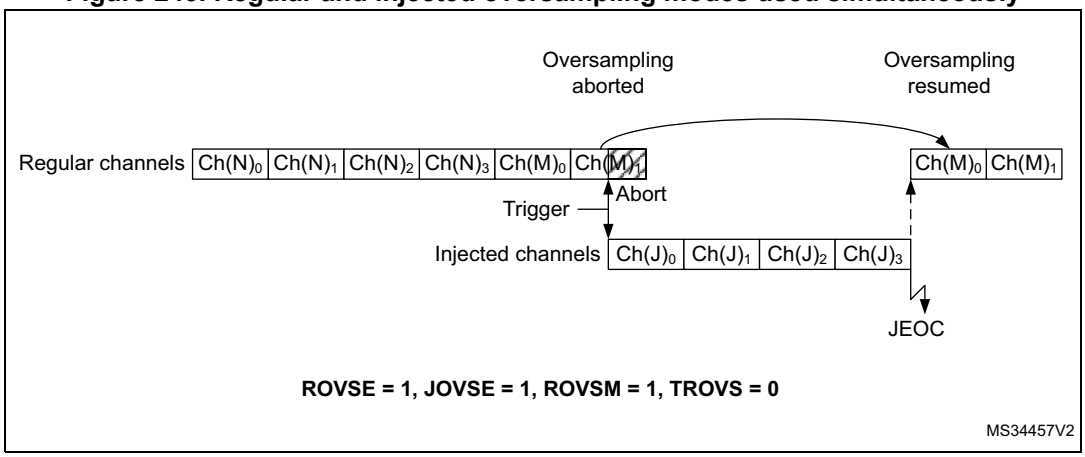
Oversampling Injected channels only

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on *Figure 243* below.

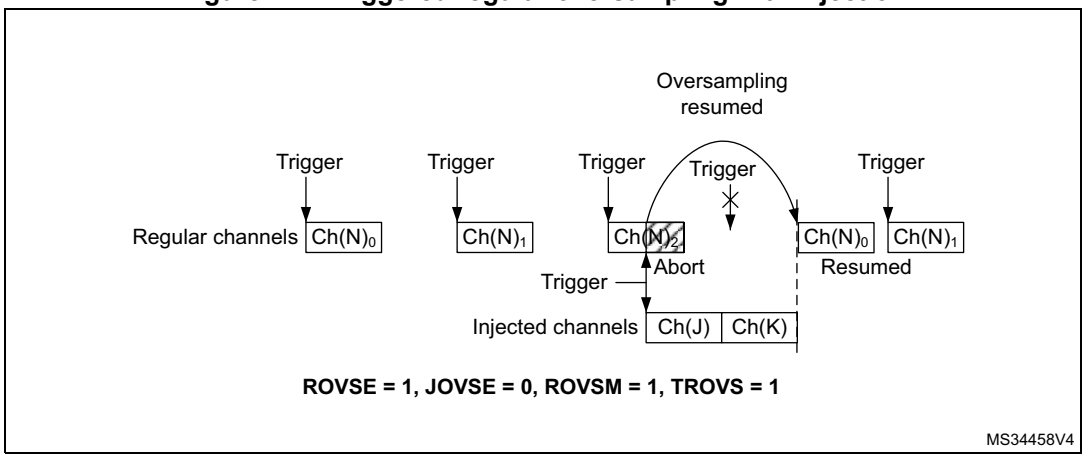
Figure 243. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on *Figure 244* below.

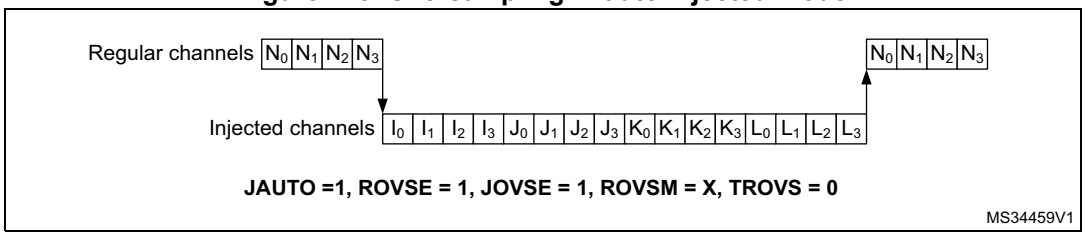
Figure 244. Triggered regular oversampling with injection



Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The *Figure 245* below shows how the conversions are sequenced.

Figure 245. Oversampling in auto-injected mode



It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

Dual ADC modes supported when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

Combined modes summary

The *Table 266* below summarizes all combinations, including modes not supported.

Table 266. Oversampler operating modes summary

Regular Oversampling ROVSE	Injected Oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered Regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

26.4.30 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 246*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADC_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADC_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

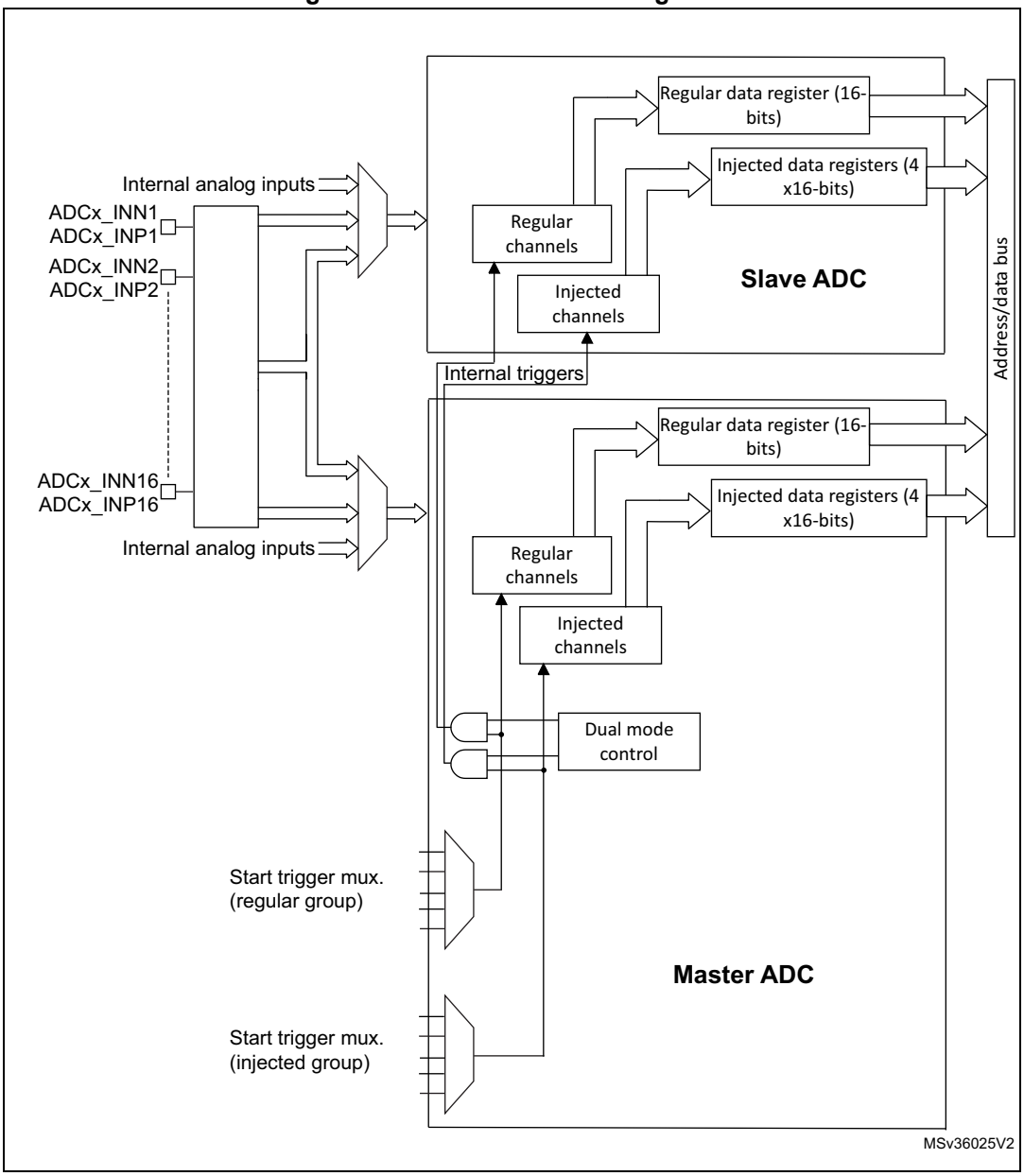
To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADC_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADC_CSR).

Figure 246. Dual ADC block diagram(1)



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADC_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

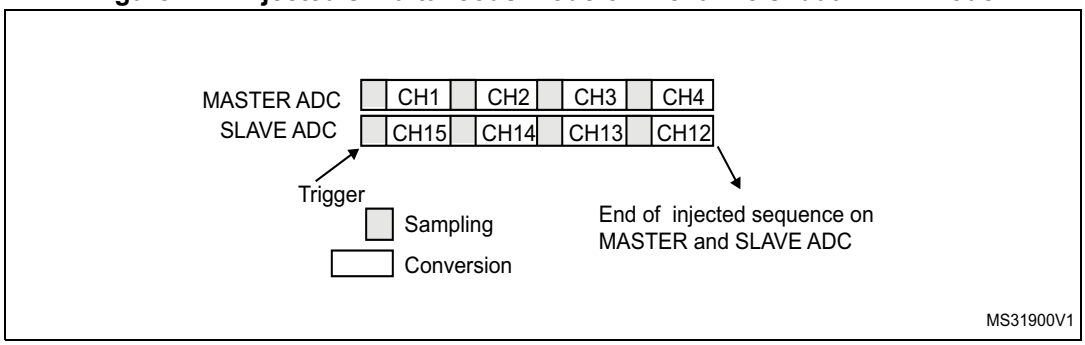
Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in *Figure 247*), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 247. Injected simultaneous mode on 4 channels: dual ADC mode



If JDISCEN = 1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the

ongoing regular sequence and the associated delay phases are ignored.
There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) aborts the current simultaneous conversions, which are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Software is notified by interrupts when it can read the data:

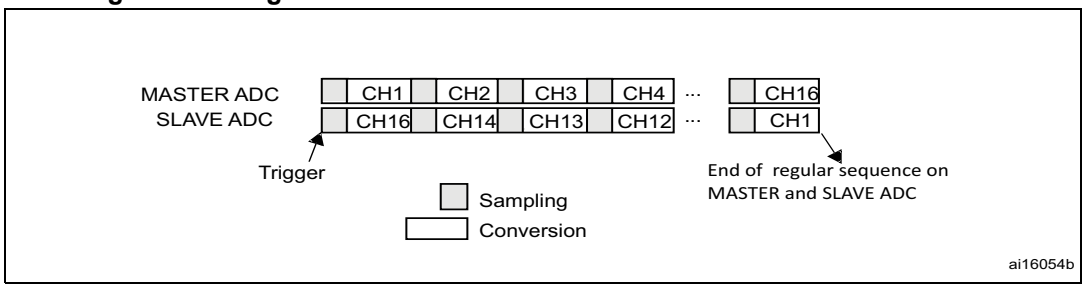
- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in *Figure 248*), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADC_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Using MDMA mode, which leaves one DMA channel free for other uses:
  - Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CDR register.
  - Both EOC flags are cleared when the DMA reads the ADC_CDR register.

Note: In MDMA mode (MDMA[1:0] = 0b10 or 0b11), the user must program the same number of conversions in the master's sequence as in the slave's sequence. Otherwise, the remaining conversions does not generate a DMA request.

Figure 248. Regular simultaneous mode on 16 channels: dual ADC mode



If DISCEN = 1 then each “n” simultaneous conversions of the regular sequence require a regular trigger event to occur (“n” is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADC_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADC_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multiple-DMA mode is used: bits MDMA must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

Interleaved mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates two conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. This delay starts counting one half cycle after the end of the sampling phase of the master conversion. This way, an ADC cannot start a

conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADC_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as following:

- Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CCR register.
- Both EOC flags are cleared when the DMA reads the ADC_CCR register.

Figure 249. Interleaved mode on one channel in continuous conversion mode: dual ADC mode

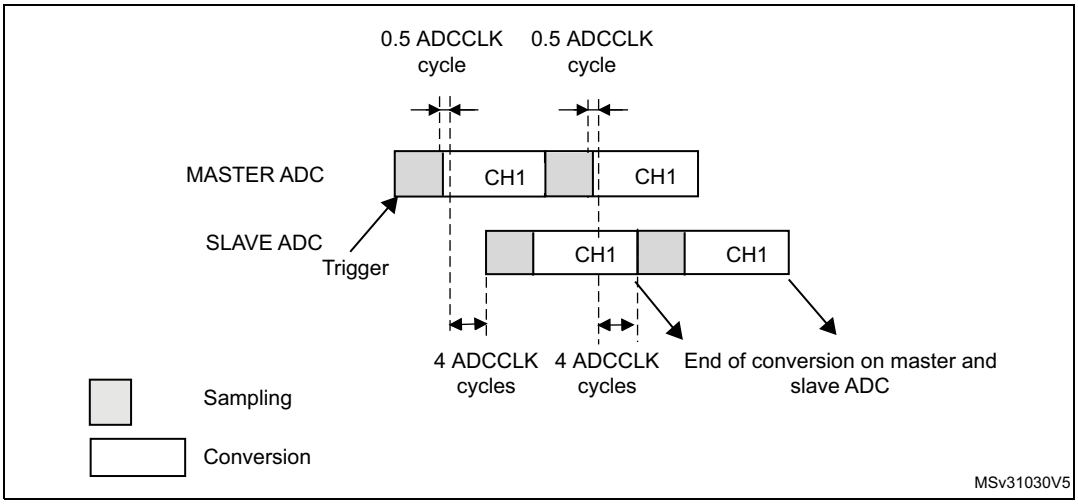
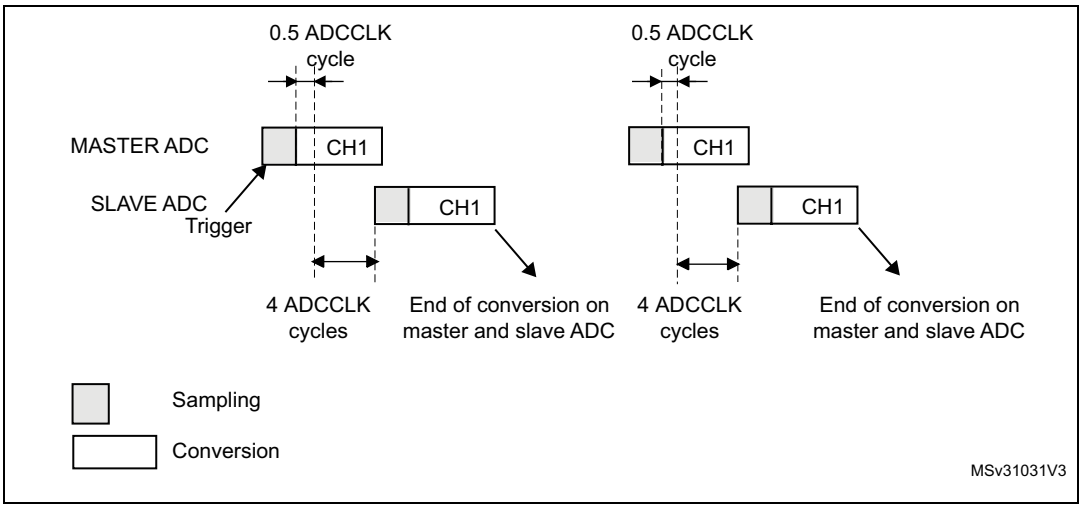


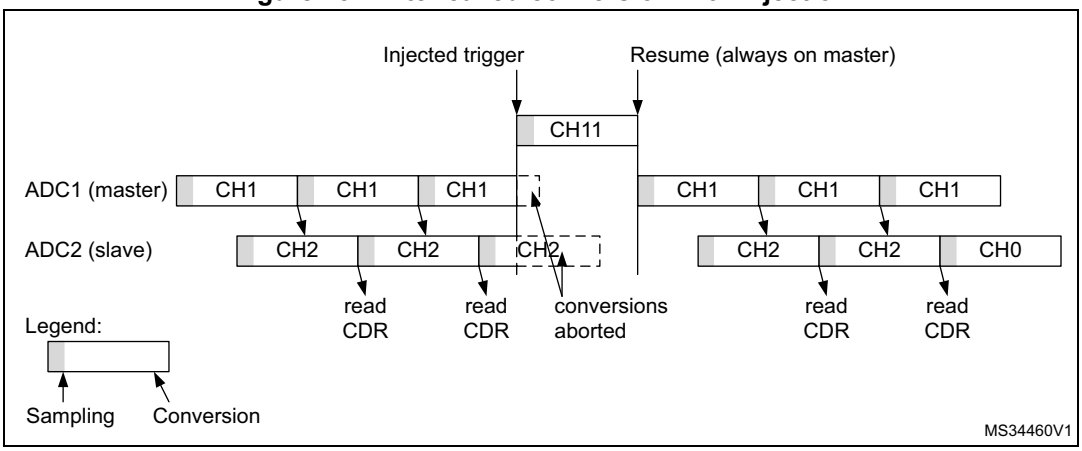
Figure 250. Interleaved mode on 1 channel in single conversion mode: dual ADC mode



If DISCEN = 1, each “n” simultaneous conversions (“n” is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 251* below).

Figure 251. Interleaved conversion with injection



Alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.

Injected discontinuous mode disabled (JDISCEN = 0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

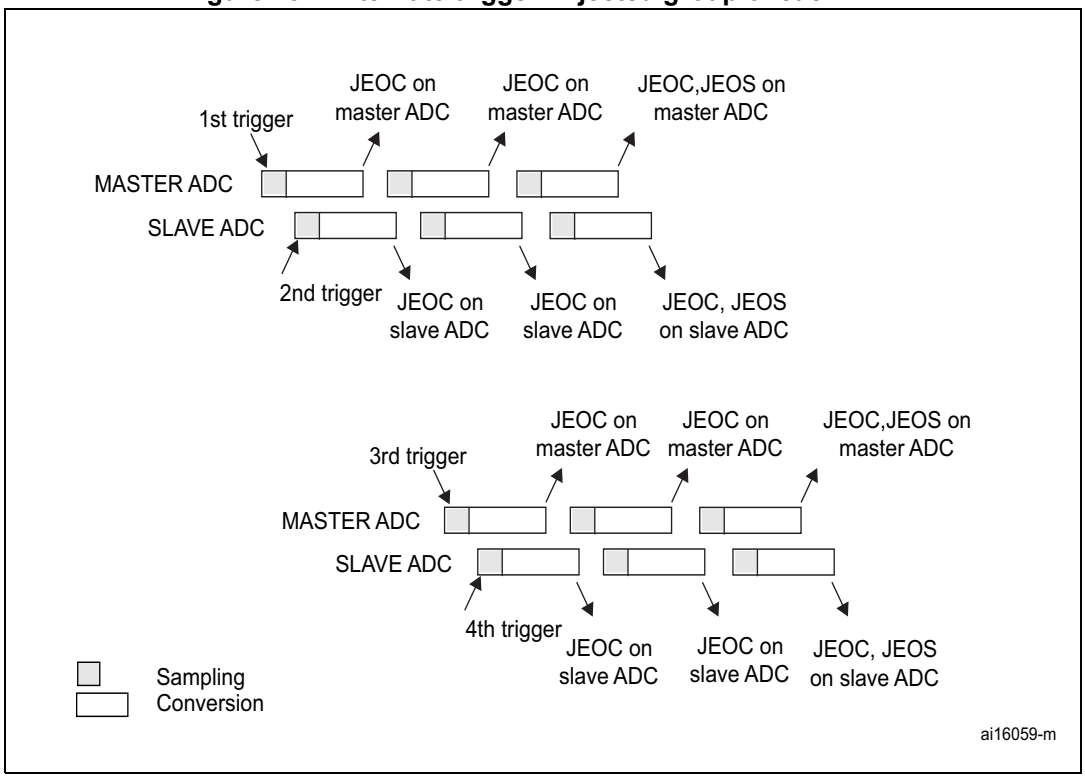
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 252. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN = 1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

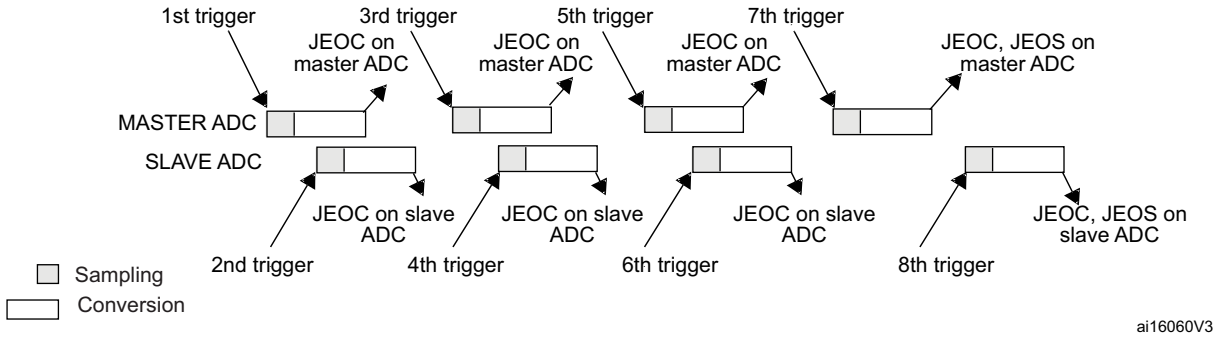
A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 253. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

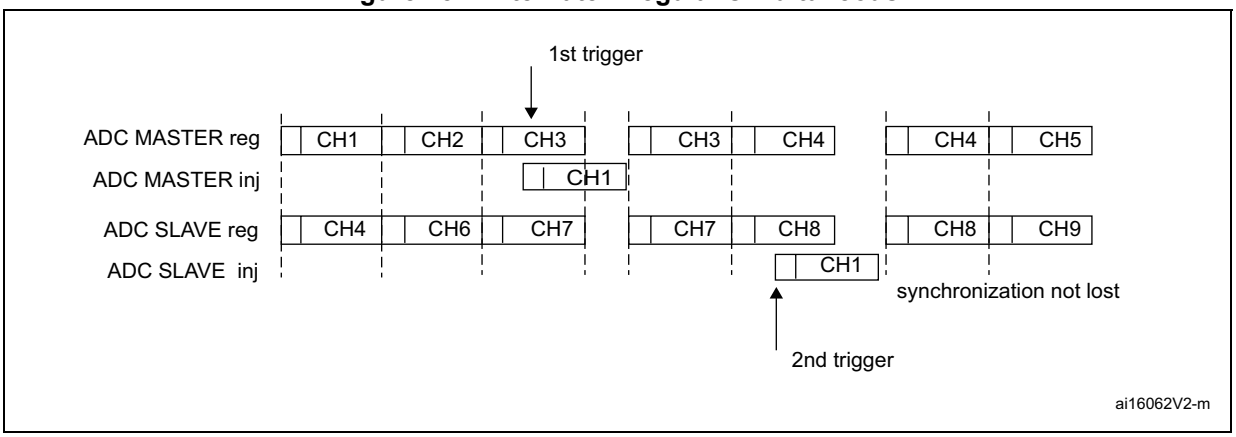
This mode is selected by programming bits DUAL[4:0] = 00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. *Figure 254* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

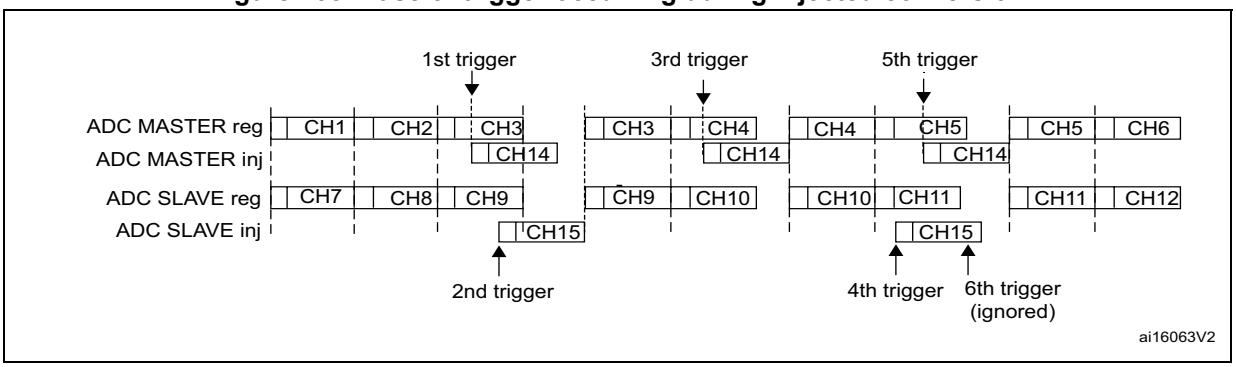
Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 254. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 255* shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 255. Case of trigger occurring during injected conversion



Combined injected simultaneous plus interleaved

This mode is selected by programming bits DUAL[4:0] = 00011

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is alway the master's one. *Figure 256*, *Figure 257* and *Figure 258* show the behavior using an example.

Caution: In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.

Figure 256. Interleaved single channel CH0 with injected sequence CH11, CH12

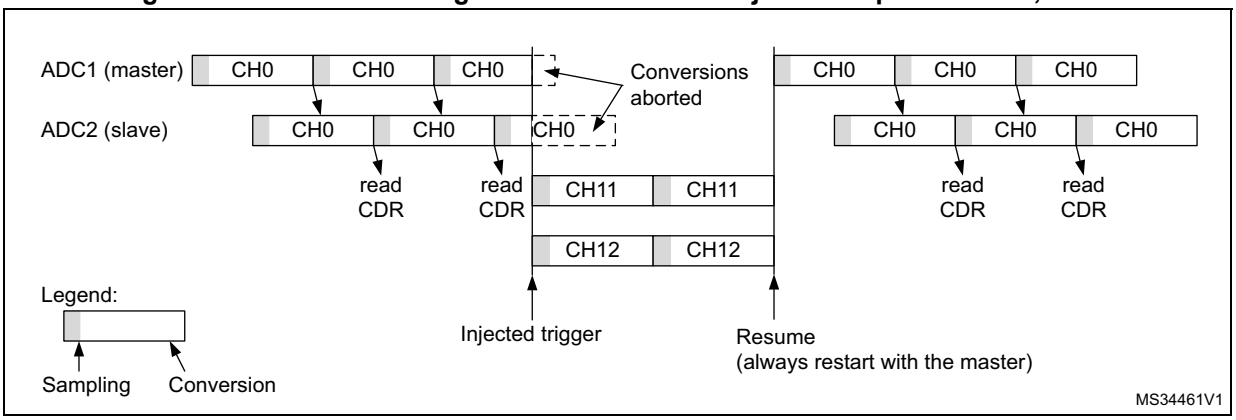


Figure 257. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

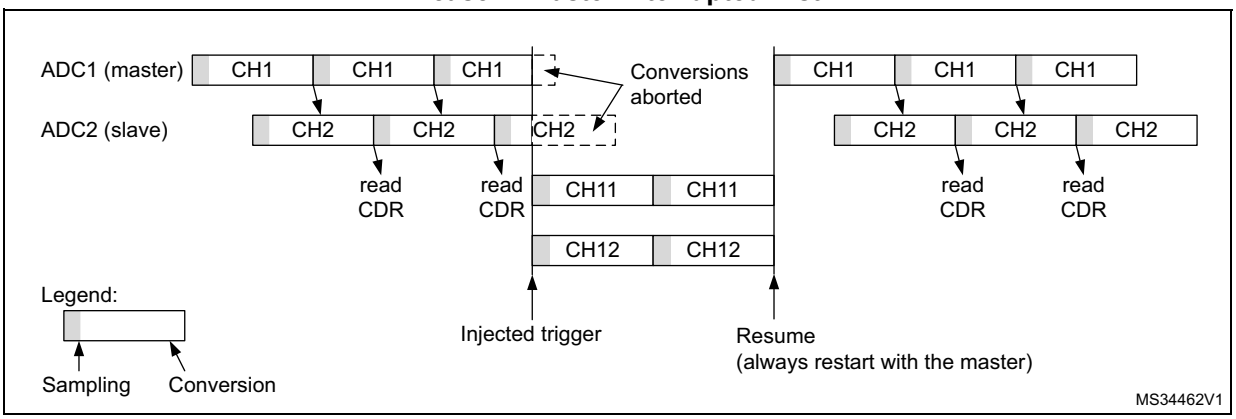
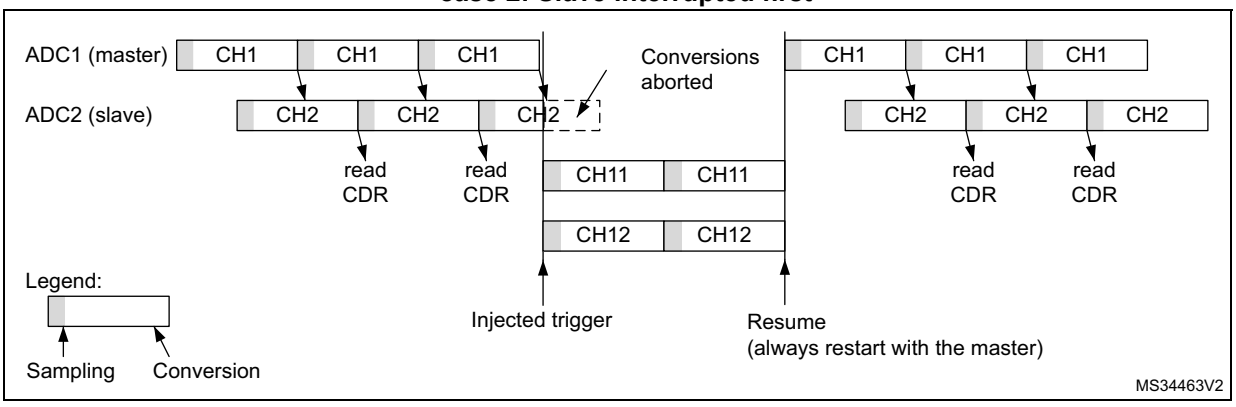


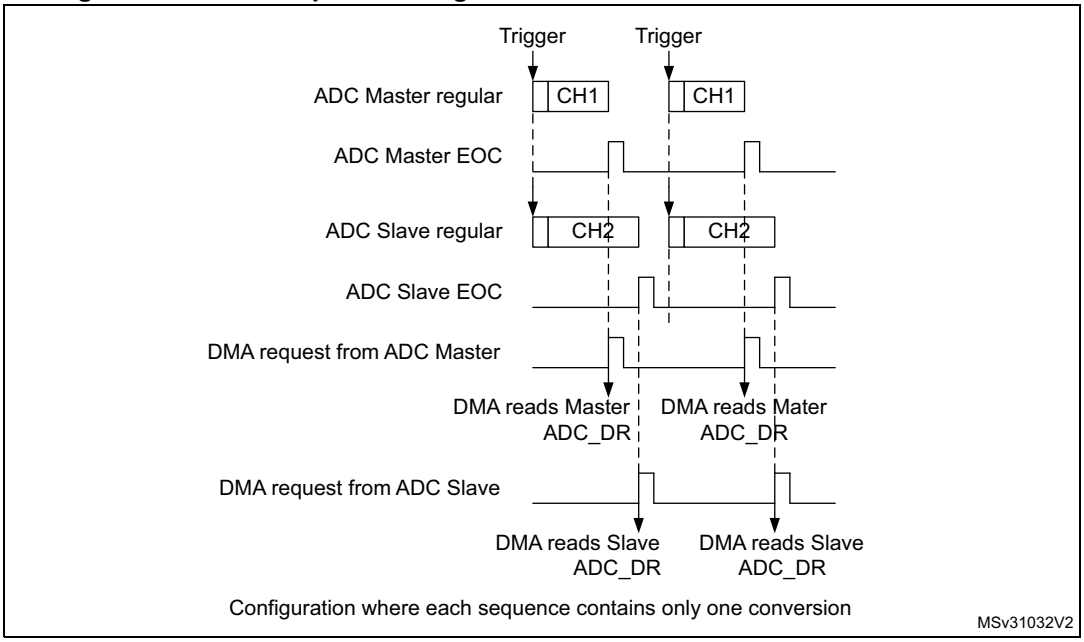
Figure 258. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 259: DMA Requests in regular simultaneous mode when MDMA = 0b00*).

Figure 259. DMA Requests in regular simultaneous mode when MDMA = 0b00



In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this MDMA bits must be configured in the ADC_CCR register:

- **MDMA = 0b10**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADC_CDR contains the two half-words representing two ADC-converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.
  This mode is used in interleaved mode and in regular simultaneous mode when resolution is 10-bit or 12-bit.

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

  2nd DMA request: ADC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 260. DMA requests in regular simultaneous mode when MDMA = 0b10

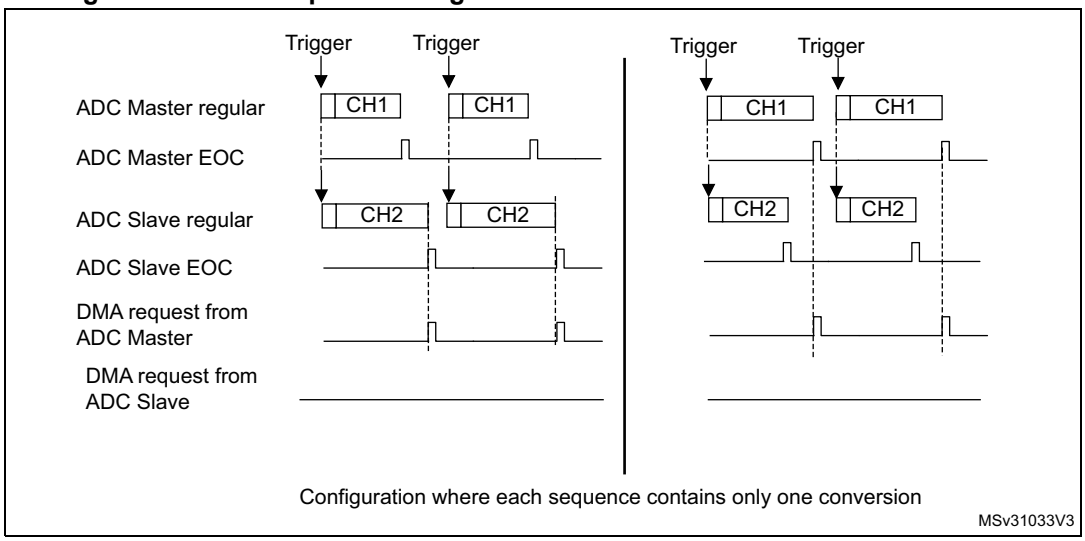
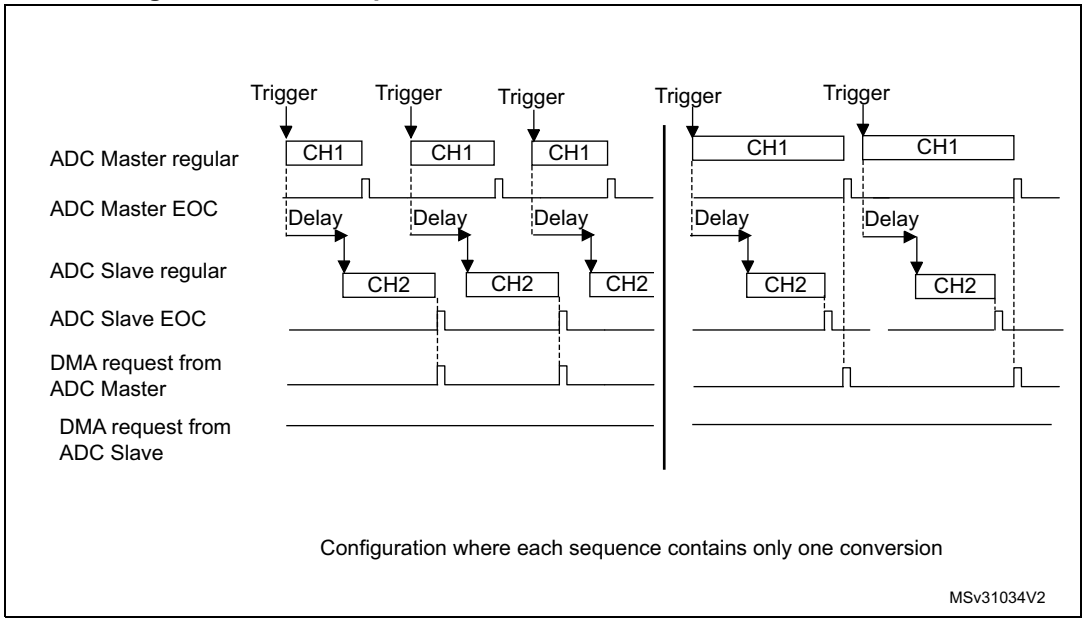


Figure 261. DMA requests in interleaved mode when MDMA = 0b10



Note: When using MDMA mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **MDMA = 0b11**: This mode is similar to the MDMA = 0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.

  This mode is used in interleaved and regular simultaneous mode when resolution is 6-bit or when resolution is 8-bit and data is not signed (offsets must be disabled for all the involved channels).

  Example:

  Interleaved dual mode: a DMA request is generated each time 2 data items are available:

  1st DMA request: ADC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

  2nd DMA request: ADC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Overrun detection

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the MDMA configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

DMA one shot mode/ DMA circular mode when MDMA mode is selected

When MDMA mode is selected (0b10 or 0b11), bit DMACFG of the ADC_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA* (bits DMACFG of master and slave ADC_CFGR are not relevant).

Stopping the conversions in dual ADC modes

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

26.4.31 Temperature sensor

The temperature sensor can be used to measure the junction temperature (Tj) of the device.

The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value (see *Table: ADC interconnection* in *Section 26.4.2: ADC pins and internal signals* for more details). When not in use, the sensor can be put in power down mode. It support the temperature range –40 to 125 °C.

Figure 262 shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).

The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

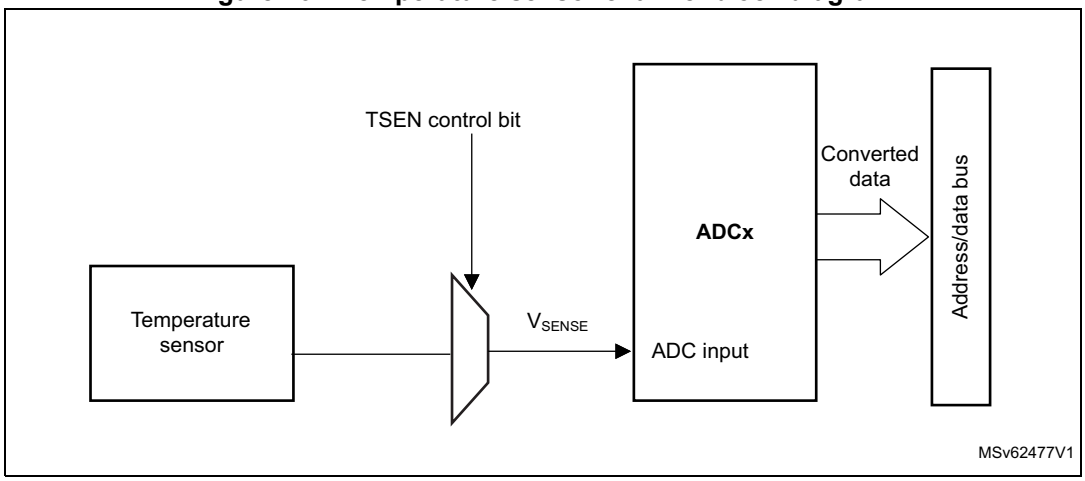
During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor's output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode.

Figure 262 shows the block diagram of the temperature sensor.

Figure 262. Temperature sensor channel block diagram



Reading the temperature

To use the sensor:

1. Select the ADC input channels that is connected to V_{SENSE}.
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the bit in the ADC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting V_{SENSE} data in the ADC data register.
6. Calculate the actual temperature using the following formula:

$$\text{Temperature (in °C)} = \frac{\text{TS_CAL2_TEMP} - \text{TS_CAL1_TEMP}}{\text{TS_CAL2} - \text{TS_CAL1}} \times (\text{TS_DATA} - \text{TS_CAL1}) + \text{TS_CAL1_TEMP}$$

where:

- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

  Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking from power-down mode before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and bits should be set at the same time.

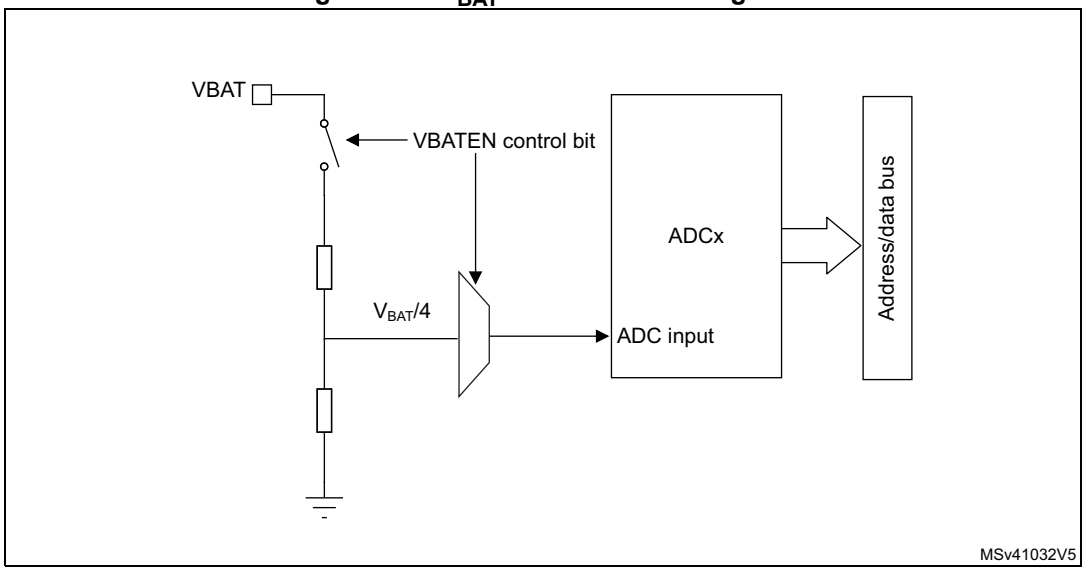
26.4.32 V_{BAT} supply monitoring

The VBATEN bit in the ADC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage could be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the ADC input channels (see *Table: ADC interconnection* in *Section 26.4.2: ADC pins and internal signals* for more details). As a consequence, the converted digital value is one third of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 263 shows the block diagram of the V_{BAT} sensing feature.

Figure 263. V_{BAT} channel block diagram



1. The VBATEN bit must be set to enable the conversion of internal channel for V_{BAT}/4.

26.4.33 Monitoring the internal voltage reference

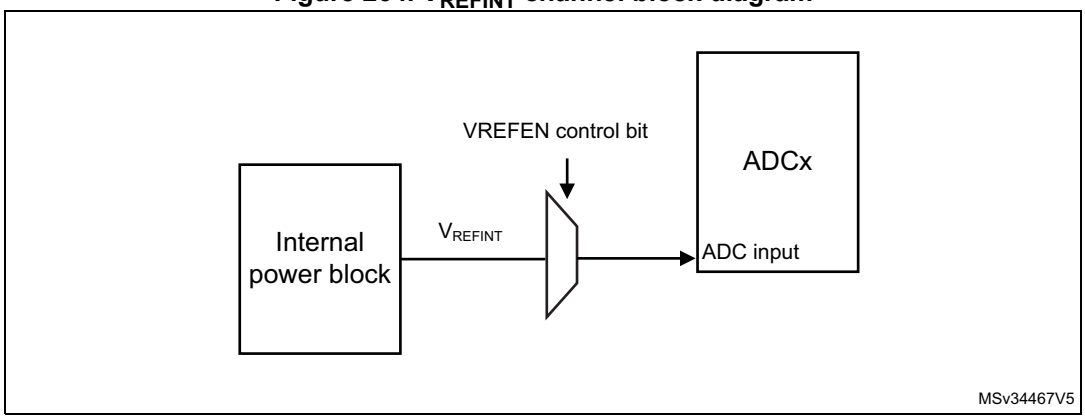
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to *Table: ADC interconnection* in *Section 26.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 264 shows the block diagram of the V_{REFINT} sensing feature.

Figure 264. V_{REFINT} channel block diagram



1. The VREFEN bit into ADC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

V_{REF+} voltage may be subject to variations or not precisely known. The embedded internal reference voltage V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+_charac} can be used to evaluate the actual V_{REF+} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINVREFINT_CAL is the VREFINT calibration value
- VREFINT_DATA is the actual VREFINT output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For applications where V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process.
- VREFINT_CAL is the VREFINT calibration value
- ADC_DATA is the value measured by the ADC on channel x (right-aligned)
- VREFINT_DATA is the actual VREFINT output value converted by the ADC
- FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, it is 2^{12} - 1 = 4095 or with 8-bit resolution, 2^8 - 1 = 255.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

26.4.34 Monitoring the supply voltage

ADC2 is connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.

26.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.

Table 267. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 status bit is set	AWD1	AWD1IE			
	Analog watchdog 2 status bit is set	AWD2	AWD2IE			
	Analog watchdog 3 status bit is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			
	Injected context queue overflows	JQOVF	JQOVFIE			

26.6 ADC registers (for each ADC)

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

26.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF**: Injected context queue overflow

This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to *Section 26.4.21: Queue of context for injected conversions* for more information.

0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)

1: Injected context queue overflow has occurred

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software. writing 1 to it.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.

0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular Conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

26.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **JQOVFIE**: Injected context queue overflow interrupt enable

This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.

0: Injected Context Queue Overflow interrupt disabled

1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

26.6.3 ADC control register (ADC_CR)

Address offset: 0x08

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or Differential inputs mode.

It is cleared by hardware after calibration is complete.

0: Calibration complete

1: Write 1 to calibrate the ADC. Read at 1 means that a calibration in progress.

Note: The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing)

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or Differential inputs mode for the calibration.

0: Writing ADCAL launches a calibration in single-ended inputs mode.

1: Writing ADCAL launches a calibration in Differential inputs mode.

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bits is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled

1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to *Section 26.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)*.

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion immediately starts (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (JEXTSEL = 0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.

– in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion immediately starts (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

– in single conversion mode when software trigger is selected (EXTSEL = 0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.

– in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)

26.6.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ALIGN	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES[1:0]		Res.	DMACFG	DMAEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **JQDIS**: Injected queue disable

This bit is set and cleared by software to disable the injected queue mechanism:

0: Injected queue enabled

1: Injected queue disabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).

A set or reset of JQDIS bit causes the injected queue to be flushed and the ADC_JSQR register is cleared.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10011: ADC analog input channel 19 monitored by AWD1

others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

The channel selected by AWD1CH must be also selected into the SQi or JSQi bits.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 **JQM**: ADC_JSQR queue mode

This bit is set and cleared by software.

It defines how an empty Queue is managed.

0: ADC_JSQR mode 0: The Queue is never empty and maintains the last written configuration into ADC_JSQR.

1: ADC_JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 26.4.21: Queue of context for injected conversions* for more information.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

When dual mode is enabled (bits DUAL of ADC_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to *Section : Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)*.

0: Right alignment

1: Left alignment

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

0: Auto-delayed conversion mode off

1: Auto-delayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0
00001: adc_ext_trg1
00010: adc_ext_trg2
00011: adc_ext_trg3
00100: adc_ext_trg4
00101: adc_ext_trg5
00110: adc_ext_trg6
00111: adc_ext_trg7
...
11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 4:3 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMACFG**: Direct memory access configuration

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

0: DMA One Shot mode selected
1: DMA Circular mode selected

For more details, refer to *Section : Managing conversions using the DMA*

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bit DMACFG of the ADC_CCR register.

Bit 0 **DMAEN**: Direct memory access enable

This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA to manage automatically the converted data. For more details, refer to *Section : Managing conversions using the DMA*.

0: DMA disabled
1: DMA enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

In dual-ADC modes, this bit is not relevant and replaced by control bits MDMA[1:0] of the ADC_CCR register.

26.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				OVSR[2:0]			JOVSE	ROVSE
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled

1: Sampling time control trigger mode enabled

The sampling time starts on the trigger rising edge, and the conversion on the trigger falling edge.

EXTEN bit should be set to 01. BULB bit must not be set when the SMPTRIG bit is set.

When EXTEN bit is set to 00, set SWTRIG to start the sampling and clear SWTRIG bit to start the conversion.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 26 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

SAMPTRIG bit must not be set when the BULB bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Software trigger starts the conversion for sampling time control trigger mode

1: Software trigger starts the sampling for sampling time control trigger mode

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 24:17 Reserved, must be kept at reset value.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 9 **TROVS**: Triggered Regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift

0001: Shift 1-bit

0010: Shift 2-bits

0011: Shift 3-bits

0100: Shift 4-bits

0101: Shift 5-bits

0110: Shift 6-bits

0111: Shift 7-bits

1000: Shift 8-bits

Other codes reserved

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:2 **OVSR[2:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

000: 2x

001: 4x

010: 8x

011: 16x

100: 32x

101: 64x

110: 128x

111: 256x

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 1 **JOVSE**: Injected oversampling Enable

This bit is set and cleared by software to enable injected oversampling.

0: Injected oversampling disabled

1: Injected oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0 **ROVSE**: Regular oversampling Enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular oversampling disabled

1: Regular oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

26.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SMPPLUS	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SMPPLUS:** Addition of one clock cycle to the sampling time.

1: 2.5 ADC clock cycle sampling time becomes 3.5 ADC clock cycles for the ADC_SMPR1 and ADC_SMPR2 registers.

0: The sampling time remains set to 2.5 ADC clock cycles remains

To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0.

Bit 30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles

001: 6.5 ADC clock cycles

010: 12.5 ADC clock cycles

011: 24.5 ADC clock cycles

100: 47.5 ADC clock cycles

101: 92.5 ADC clock cycles

110: 247.5 ADC clock cycles

111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

26.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 2.5 ADC clock cycles
001: 6.5 ADC clock cycles
010: 12.5 ADC clock cycles
011: 24.5 ADC clock cycles
100: 47.5 ADC clock cycles
101: 92.5 ADC clock cycles
110: 247.5 ADC clock cycles
111: 640.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

26.6.8 ADC watchdog threshold register 1 (ADC_TR1)

Address offset: 0x20

Reset value: 0x0FFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HT1[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AWDFILT[2:0]			LT1[11:0]											
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to *Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **AWDFILT[2:0]**: Analog watchdog filtering parameter

This bit is set and cleared by software.

000: No filtering

001: two consecutive detection generates an AWDx flag or an interrupt

...

111: Eight consecutive detection generates an AWDx flag or an interrupt

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 11:0 **LT1[11:0]**: Analog watchdog 1 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 1.

Refer to *Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

26.6.9 ADC watchdog threshold register 2 (ADC_TR2)

Address offset: 0x24

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT2[7:0]**: Analog watchdog 2 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 2.

Refer to *Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT2[7:0]**: Analog watchdog 2 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 2.

Refer to *Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

26.6.10 ADC watchdog threshold register 3 (ADC_TR3)

Address offset: 0x28

Reset value: 0x00FF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 3.

Refer to *Section 26.4.28: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)*

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold

These bits are written by software to define the lower threshold for the analog watchdog 3.

This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

26.6.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	L[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.

0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.12 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 8th in the regular conversion sequence

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 7th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 6th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 5th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 14th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 13th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 12th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 11th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 10th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.15 ADC regular data register (ADC_DR)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RDATA[15:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 26.4.26: Data management*.

26.6.16 ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JL[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS = 0 (queue enabled), hardware and software trigger detection disabled. Otherwise, the queue is disabled as well as hardware trigger detection (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 26.4.21: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JL[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

26.6.17 ADC offset y register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

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

Bit 31 **OFFSET_EN:** Offset y enable

This bit is written by software to enable or disable the offset programmed into bits OFFSET[11:0].

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 30:26 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed into bits OFFSET[11:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the data offset y.

If OFFSET_EN is set, it is not allowed to select the same channel for different ADC_OFRy registers.

Bit 25 **SATEN**: Saturation enable

This bit is set and cleared by software to enable the saturation at 0x000 and 0xFFF for the offset function.

0: No saturation control, offset result can be signed

1: Saturation enabled, offset result unsigned and saturated at 0x000 and 0xFFF

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **OFFSETPOS**: Positive offset

This bit is set and cleared by software to enable the positive offset.

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET[11:0]:** Data offset y for the channel programmed into bits OFFSET_CH[4:0]

These bits are written by software to define the offset to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset must be programmed in the bits OFFSET_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

If several offset (OFFSET) point to the same channel, only the offset with the lowest x value is considered for the subtraction.
Ex: if OFFSET1_CH[4:0] = 4 and OFFSET2_CH[4:0] = 4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

26.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **JDATA[15:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in *Section 26.4.26: Data management*.

26.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0xA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD2CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD2CH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWD2CH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWD2CH must be also selected into the SQi or JSQi bits.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

26.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AWD3CH[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWD3CH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWD3CH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWD3CH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWD3CH[19:0] = 000..0, the analog Watchdog 3 is disabled

Note: The channels selected by AWD3CH must be also selected into the SQi or JSQi bits.

The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Some channels are not connected physically and must not be selected for the analog watchdog.

26.6.21 ADC Differential mode selection register (ADC_DIFSEL)

Address offset: 0xB0

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]:** Differential mode for channels 19 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or Differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in Differential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

26.6.22 ADC calibration factors (ADC_CALFACT)

Address offset: 0xB4

Reset value: 0x0000 0000

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

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

Bits 22:16 **CALFACT_D[6:0]**: Calibration Factors in differential mode

These bits are written by hardware or by software.

Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new differential calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

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

Bits 6:0 **CALFACT_S[6:0]**: Calibration Factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.

Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

26.6.23 ADC option register (ADC_OR)

Address offset: 0xC8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP0
															rw

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

Bit 0 **OP0**: Option bit 0

For ADC1:

0: INP0/INN1 GPIO switch control disabled

1: INP0/INN1 GPIO switch control enabled

Note: This option bit must be set to 1 when ADCx_INP0 or ADCx_INN1 channel is selected.

For ADC2:

0: V_{DDCORE} channel disabled

1: V_{DDCORE} channel enabled

Note: ADC_OR register might be reserved on some ADC instances. Refer to Section 26.3: ADC implementation.

26.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs:

26.7.1 ADC common status register (ADC_CSR)

Address offset: 0x300

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADC. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **JQOVF_SLV:** Injected Context Queue Overflow flag of the slave ADC

This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC

This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC

This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC

This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC

This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **JQOVF_MST:** Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

26.7.2 ADC common control register (ADC_CCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	PRESC[3:0]				CKMODE[1:0]	
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDMA[1:0]		DMACFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable

This bit is set and cleared by software to control.

0: V_{BAT} channel disabled

1: V_{BAT} channel enabled

Bit 23 **TSEN**: V_{SENSE} enable

This bit is set and cleared by software to control V_{SENSE}.

0: Temperature sensor channel disabled

1: Temperature sensor channel enabled

Bit 22 **VREFEN**: V_{REFINT} enable

This bit is set and cleared by software to enable/disable the V_{REFINT} channel.

0: V_{REFINT} channel disabled

1: V_{REFINT} channel enabled

Bits 21:18 **PRESC[3:0]**: ADC prescaler

These bits are set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.

0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
other: reserved

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.

Bits 17:16 **CKMODE[1:0]:** ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

00: adc_ker_ck (x = 1/2) (Asynchronous clock mode), generated at product level (refer to *Section 6: Reset and clock control (RCC)*)

01: adc_hclk/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0XXX in RCC_CFGR register) and if the system clock has a 50% duty cycle.

10: adc_hclk/2 (Synchronous clock mode)

11: adc_hclk/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 15:14 **MDMA[1:0]:** Direct memory access mode for dual ADC mode

This bitfield is set and cleared by software. Refer to the DMA controller section for more details.

00: MDMA mode disabled

01: Reserved

10: MDMA mode enabled for 12 and 10-bit resolution

11: MDMA mode enabled for 8 and 6-bit resolution

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 13 **DMACFG**: DMA configuration (for dual ADC mode)

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

0: DMA One Shot mode selected

1: DMA Circular mode selected

For more details, refer to *Section : Managing conversions using the DMA*

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 268* for the value of ADC resolution versus DELAY bits values.

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection

These bits are written by software to select the operating mode. 00000 corresponds to Independent mode. Values 00001 to 01001 correspond to dual mode, master and slave ADCs working together.

00000: Independent mode
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Combined interleaved mode + injected simultaneous mode
00100: Reserved
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
Others: Reserved, must not be used

Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Table 268. DELAY bits versus ADC resolution

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1010	11 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
others	12 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$

26.7.3 ADC common regular data register for dual mode (ADC_CDR)

Address offset: 0x30C

Reset value: 0x0000 0000

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

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 26.4.30: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 26.4.30: Dual ADC modes*.

The data alignment is applied as described in *Section : Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)*)

In MDMA = 0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

26.8 ADC register map

The following table summarizes the ADC registers.

Table 269. ADC global register map

Offset	Register
0x000 - 0x0B4	Master ADC1
0x0B8 - 0x0FC	Reserved
0x100 - 0x1B4	Slave ADC2
0x1B8 - 0x2FC	Reserved
0x300 - 0x30C	Master and slave ADCs common registers

Table 270. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**ADC_ISR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVF	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x04	**ADC_IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JQOVFIE	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x08	**ADC_CR**	ADCAL	ADCALDIF	DEEPPWD	ADVREGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1	0																							0	0	0	0	0	0
0x0C	**ADC_CFGR**	JQDIS	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	JQM	JDISCEN	DISCNUM[2:0]			DISCEN	ALIGN	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL4	EXTSEL3	EXTSEL2	EXTSEL1	EXTSEL0	RES[1:0]		Res.	DMACFG	DMAEN
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x0C	**ADC_CFGR2**	Res.	Res.	Res.	Res.	SMPTRIG	BULB	SWTRIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROVSM	TROVS	OVSS[3:0]					OVSR[2:0]			JOVSE	ROVSE
	Reset value					0	0	0															0	0	0	0	0	0	0	0	0	0	0
0x14	**ADC_SMPR1**	SMPPLUS	Res.	SMP9 [2:0]			SMP8 [2:0]			SMP7 [2:0]			SMP6 [2:0]			SMP5 [2:0]			SMP4 [2:0]			SMP3 [2:0]			SMP2 [2:0]			SMP1 [2:0]			SMP0 [2:0]		
	Reset value	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**ADC_SMPR2**	Res.	Res.	SMP19 [2:0]			SMP18 [2:0]			SMP17 [2:0]			SMP16 [2:0]			SMP15 [2:0]			SMP14 [2:0]			SMP13 [2:0]			SMP12 [2:0]			SMP11 [2:0]			SMP10 [2:0]		
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	Reserved	Res.																															
0x20	**ADC_TR1**	Res.	Res.	Res.	Res.	HT1[11:0]												Res.	AWDFILT [2:0]			LT1[11:0]											
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 270. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24	**ADC_TR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT2[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT2[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x28	**ADC_TR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HT3[[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LT3[7:0]							
	Reset value									1	1	1	1	1	1	1	1									0	0	0	0	0	0	0	0
0x2C	Reserved	Res.																															
0x30	**ADC_SQR1**	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]			Res.	Res.	L[3:0]					
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x34	**ADC_SQR2**	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]			Res.	SQ5[4:0]						
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x38	**ADC_SQR3**	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]			Res.	SQ10[4:0]						
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x3C	**ADC_SQR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]			Res.	SQ15[4:0]						
	Reset value																						0	0	0	0	0		0	0	0	0	0
0x40	**ADC_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	regular RDATA[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44-0x48	Reserved	Res.																															
0x4C	**ADC_JSQR**	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]				Res.	JSQ1[4:0]				JEXTEN[1:0]		JEXTSEL [4:0]					JL[1:0]		
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50-0x5C	Reserved	Res.																															
0x60	**ADC_OFR1**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x64	**ADC_OFR2**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x68	**ADC_OFR3**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x6C	**ADC_OFR4**	OFFSET_EN	OFFSET_ CH[4:0]					SATEN	OFFSETPOS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[11:0]											
	Reset value	0	0	0	0	0	0	0	0													0	0	0	0	0	0	0	0	0	0	0	0
0x70-0x7C	Reserved	Res.																															

Table 270. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80	**ADC_JDR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x84	**ADC_JDR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	**ADC_JDR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8C	**ADC_JDR4**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JDATA4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90-0x9C	Reserved	Res.																															
0xA0	**ADC_AWD2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD2CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4	**ADC_AWD3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3CH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8-0xAC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xB0	**ADC_DIFSEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB4	**ADC_CALFACT**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[6:0]						
	Reset value										0	0	0	0	0	0	0										0	0	0	0	0	0	0
0xB8-0xC4	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0xC8	**ADC_OR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP0
	Reset value																																0
0xCC-0xFC	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 271. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	**ADC_CSR**	Res.	Res.	Res.	Res.	Res.	JQOVF_SLV	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	JQOVF_MST	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
							slave ADC2																master ADC1										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x304	Reserved	Res.																															
0x308	**ADC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	TSEN	VREFEN	PRESC[3:0]				CKMODE[1:0]		MDMA[1:0]		DMACFG	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0				0	0	0	0	0

Table 271. ADC register map and reset values (master and slave ADC common registers) (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30C	**ADC_CDR**	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x310-0x3EC	Reserved	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res	res

Refer to *Section 2.3 on page 115* for the register boundary addresses.

27 Digital temperature sensor (DTS)

27.1 DTS introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

27.2 DTS main features

The temperature sensor block main features are the following:

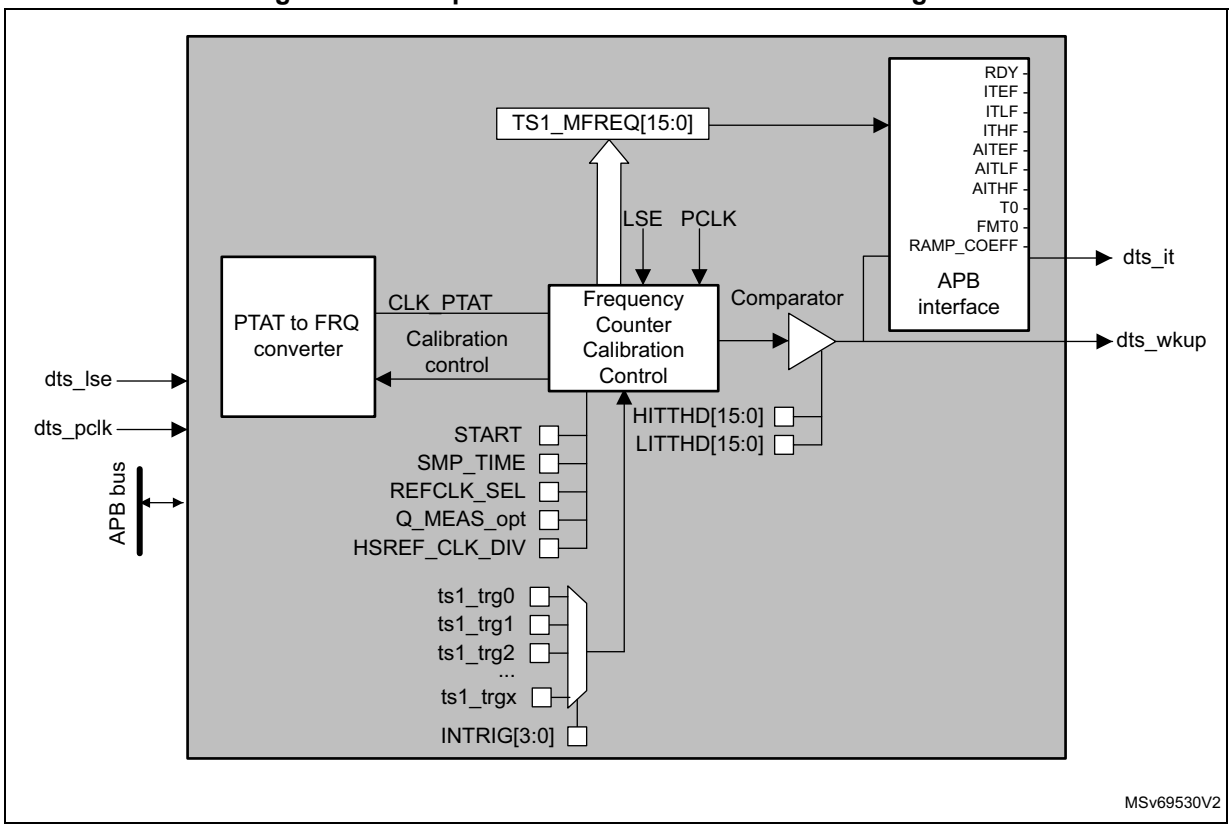
- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wakeup signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
- Quick measurement using LSE clock

27.3 DTS functional description

27.3.1 DTS block diagram

The temperature sensor block diagram is shown in *Figure 265*.

Figure 265. Temperature sensor functional block diagram



27.3.2 DTS internal signals

Table 272. DTS internal input/output signals

Signal name	Signal type	Description
dts_lse	Digital input	LSE clock
dts_pclk	Digital input	APB clock
dts_it	Digital output	Temperature sensor interrupt
dts_wkup	Digital output	Temperature sensor wakeup

27.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

27.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in *Temperature sensor configuration register 1 (DTS_CFGR1)*:

- PCLK only (REFCLK_SEL = 0)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock
- PCLK and LSE (REFCLK_SEL = 1)

  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.
- LSE only (REFCLK_SEL = 1) and PCLK OFF

  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

27.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered except for quick measurement mode (Q_MEAS_OPT set to 1 in DTS_CFGR1).

27.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).
- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.

27.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent.

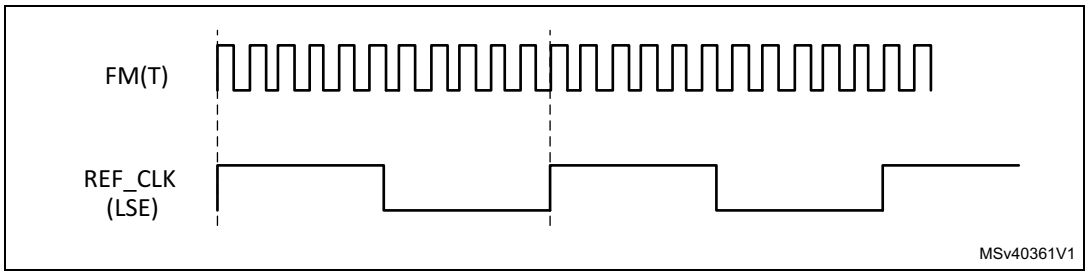
Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFGR1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

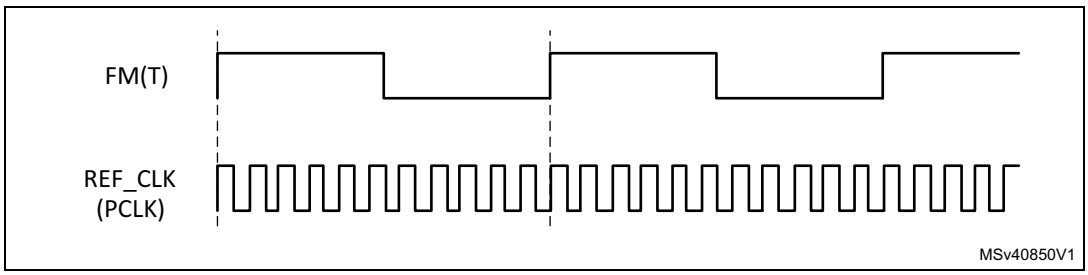
This counter behavior is shown in *Figure 266* and *Figure 267*.

Figure 266. Method for low REF_CLK frequencies



1. To increase the precision, FM(T) measurement can be done on several LSE periods.

Figure 267. Method for high REF_CLK frequencies



1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see *Temperature sensor data register (DTS_DR)*.

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

$$T = T_0 + ((F_{PCLK} / TS1_MFREQ) \times TS1_SMP_TIME - 100 \times TS1_FMT0) / TS1_RAMP_COEFF$$

where

T0 (factory calibration temperature) is equal to 30 °C.

TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.

TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

$$T = T_0 + ((F_{LSE} \times TS1_MFREQ / TS1_SMP_TIME) - (100 \times TS1_FMT0)) / TS1_RAMP_COEFF$$

27.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFGR1 register (see *Table 273*).

Table 273. Sampling time configuration

TS1_SMP_TIME[3:0]	LSE or FM(T) clock cycle(s)
0000	1
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

27.3.9 Quick measurement mode

If a high precision is not required, the calibration step included in each measurement sequence can be skipped by setting Q_MEAS_OPT to 1 in the DTS_CFGR1 register. This method must used only when the LSE clock is selected as reference clock (LSREF_CLK set to 1). This mode can reduce the measurement time.

27.3.10 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFGR1.

Table 274. Trigger configuration

Name	TS1_INTRIG[3:0]				Comment
N.A	0	0	0	0	No hardware trigger
ts1_trg0	0	0	0	1	lptim1_ch1
ts1_trg1	0	0	1	0	lptim2_ch1
ts1_trg2	0	0	1	1	lptim3_ch1
ts1_trg3	0	1	0	0	exti13
ts1_trg4	0	1	0	1	Reserved
ts1_trg5	0	1	1	0	
ts1_trg6	0	1	1	1	
ts1_trg7	1	0	0	0	
ts1_trg8	1	0	0	1	
ts1_trg9	1	0	1	0	
ts1_trg10	1	0	1	1	
ts1_trg11	1	1	0	0	
ts1_trg12	1	1	0	1	
ts1_trg13	1	1	1	0	
ts1_trg14	1	1	1	1	

Note: Hardware triggers are active only on the rising edge.

The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 27.3.11: On-off control and ready flag), otherwise the trigger is ignored.

If a trigger source changes on-the-fly, the new trigger source signal should be low. If the new source signal is high, the temperature sensor detects a rising edge and start the measurement sequence.

27.3.11 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFGR1 register. The TS1_RDY flag in the *Temperature sensor status register (DTS_SR)* indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests are then ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.

27.3.12 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

Software trigger

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to ‘0000’ in DTS_CFGR1.

If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFGR1 starts the measurement.

If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.

If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag become 1 and the measurement restarts.

Hardware trigger

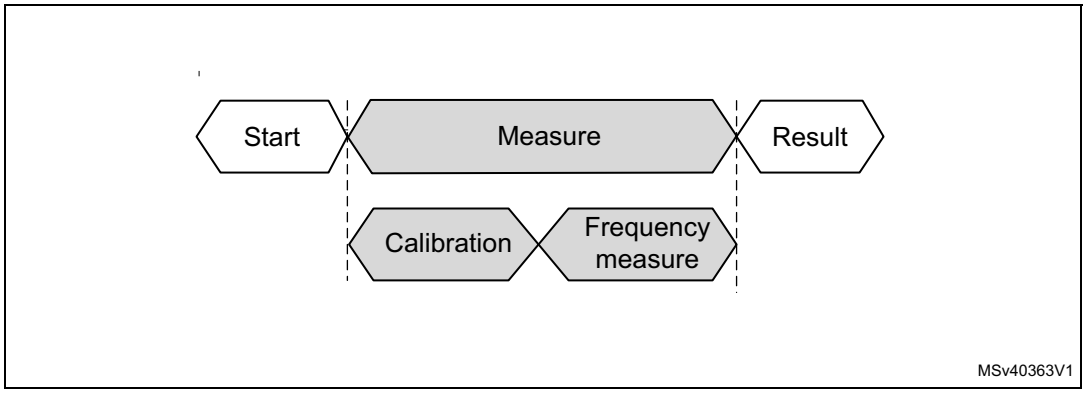
TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

Temperature measurement sequence

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see *Section 27.3.5: Calibration*). The measurement period depends on the following DTS_CFGR1 bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

Figure 268. Temperature sensor sequence



27.4 DTS low-power modes

Table 275. Temperature sensor behavior in low-power modes

Mode	Description
Sleep	Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.
Stop	Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.

27.5 DTS interrupts

There are two ways to use the DTS block as an interrupt source.The DTS interrupt line can be connected to the EXTI controller (see *Section 27.5.3: Asynchronous wakeup*) or to the CPU NVIC (see *Section 27.5.2: Synchronous interrupt*).

27.5.1 Temperature window comparator

The DTS_ITR1 register allows defining the high and low threshold that are used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ ITLF, TS1_ITHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see *Section 27.6.6*).

27.5.2 Synchronous interrupt

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 27.5.1: Temperature window comparator*).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see *Section 27.6.7*). All combinations of interrupts are allowed.

The TS1_ITEF, TS1_ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see *Section 27.6.8*).

27.5.3 Asynchronous wakeup

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 27.5.1: Temperature window comparator*).

Three asynchronous wakeup events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.

The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits an the their description.

Table 276. Interrupt control bits

Interrupt event	Interrupt flag	Enable control bit	Interrupt clear bit	Exit from Sleep mode	Synchronous/ Asynchronous
At the end of measurement	TS1_ITEF in DTS_SR	TS1_ITEEN in DTS_ITENR	TS1_CITEF in DTS_ICIFR	NO	Synchronous on PCLK
When the measure is equal or exceeds the low threshold	TS1_ITLF in DTS_SR	TS1_ITLEN in DTS_ITENR	TS1_CITLF in DTS_ICIFR	NO	
When the measure is equal or exceeds the high threshold	TS1_ITHF in DTS_SR	TS1_ITHEN in DTS_ITENR	TS1_CITHF in DTS_ICIFR	NO	
At the end of measurement	TS1_AITEF in DTS_SR	TS1_AITEEN in DTS_ITENR	TS1_CAITEF in DTS_ICIFR	YES	Asynchronous
When the measure is equal or exceeds the low threshold	TS1_AITLF in DTS_SR	TS1_AITLEN in DTS_ITENR	TS1_CAITLF in DTS_ICIFR	YES	
When the measure is equal or exceeds the high threshold	TS1_AITHF in DTS_SR	TS1_AITHEN in DTS_ITENR	TS1_CAITHF in DTS_ICIFR	YES	

27.6 DTS registers

The registers of this peripheral can only be accessed by-word (32-bit).

27.6.1 Temperature sensor configuration register 1 (DTS_CFGR1)

DTS_CFGR1 is the configuration register for temperature sensor 1.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HSREF_CLK_DIV[6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_SMP_TIME[3:0]			
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TS1_INTRIG_SEL[3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
				rw	rw	rw	rw				rw				rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSREF_CLK_DIV[6:0]**: High speed clock division ratio

These bits are set and cleared by software. They can be used to define the division ratio for the main clock in order to obtain the internal frequency lower than 1 MHz required for the calibration. They are applicable only for calibration when PCLK is selected as reference clock (REFCLK_SEL=0).

0000000: No divider

0000001: No divider

0000010: 1/2 division ratio

...

1111111: 1/127 division ratio

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **Q_MEAS_OPT**: Quick measurement option bit

This bit is set and cleared by software. It is used to increase the measurement speed by suppressing the calibration step. It is effective only when the LSE clock is used as reference clock (REFCLK_SEL=1).

0: Measurement with calibration

1: Measurement without calibration

Bit 20 **REFCLK_SEL**: Reference clock selection bit

This bit is set and cleared by software. It indicates whether the reference clock is the high speed clock (PCLK) or the low speed clock (LSE).

0: High speed reference clock (PCLK)

1: Low speed reference clock (LSE)

Bits 19:16 **TS1_SMP_TIME[3:0]**: Sampling time for temperature sensor 1

These bits allow increasing the sampling time to improve measurement precision.

When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement is performed at TS1_SMP_TIME period of CLK_PTAT.

When the LSE is selected as reference clock (REFCLK_SEL = 1), the measurement is performed at TS1_SMP_TIME period of LSE.

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

Bits 11:8 **TS1_INTRIG_SEL[3:0]**: Input trigger selection bit for temperature sensor 1

These bits are set and cleared by software. They select which input triggers a temperature measurement. Refer to *Section 27.3.10: Trigger input*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TS1_START:** Start frequency measurement on temperature sensor 1

This bit is set and cleared by software.

0: No software trigger.

1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TS1_EN:** Temperature sensor 1 enable bit

This bit is set and cleared by software.

0: Temperature sensor 1 disabled

1: Temperature sensor 1 enabled

Note: Once enabled, the temperature sensor is active after a specific delay time. The TS1_RDY flag is set when the sensor is ready.

27.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The reset value is factory trimmed.

Address offset: 0x08

Reset value: 0x000X XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_FMT0[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.
00: 30 °C
01: 130 °C
Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1
This value is expressed in 0.1 kHz.

27.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)

The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The reset value is factory trimmed.

Address offset: 0x10

Reset value: 0xXXXX XXXX

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_RAMP_COEFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_RAMP_COEFF[15:0]**: Engineering value of the ramp coefficient for the temperature sensor 1.
This value is expressed in Hz/°C.

27.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TS1_HITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_LITTHD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TS1_HITTHD[15:0]:** High interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 **TS1_LITTHD[15:0]:** Low interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

27.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_MFREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TS1_MFREQ[15:0]**: Value of the counter output value for temperature sensor 1

27.6.6 Temperature sensor status register (DTS_SR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
r									r	r	r		r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TS1_RDY:** Temperature sensor 1 ready flag
This bit is set and reset by hardware.
It indicates that a measurement is ongoing.
0: Temperature sensor 1 busy
1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHF:** Asynchronous interrupt flag for high threshold on temperature sensor 1
This bit is set by hardware when the high threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITHFEN bit is set

Bit 5 **TS1_AITLF:** Asynchronous interrupt flag for low threshold on temperature sensor 1
This bit is set by hardware when the low threshold is reached.
It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_AITLFEN bit is set

Bit 4 **TS1_AITEF:** Asynchronous interrupt flag for end of measure on temperature sensor 1
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
0: End of measure not detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_AITEFEN bit is set

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHF:** Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK
This bit is set by hardware when the high threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.
0: High threshold not reached on temperature sensor 1
1: High threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITHFEN bit is set

Bit 1 **TS1_ITLF:** Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when the low threshold is set and reached.
It is cleared by software by writing 1 to the TS1_CITLF bit in the DTS_ICIFR register.
0: Low threshold not reached on temperature sensor 1
1: Low threshold reached on temperature sensor 1
Note: This bit is active only when the TS1_ITLFEN bit is set

Bit 0 **TS1_ITEF:** Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.
This bit is set by hardware when a temperature measure is done.
It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.
0: No end of measurement detected on temperature sensor 1
1: End of measure detected on temperature sensor 1
Note: This bit is active only when the TS1_ITEFEN bit is set

27.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ AITHEN	TS1_ AITLEN	TS1_ AITEEN	Res.	TS1_ ITHEN	TS1_ ITLEN	TS1_ ITEEN
									rw	rw	rw		rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHEN:** Asynchronous interrupt enable flag on high threshold for temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is above the high threshold (only when REFCLK_SEL= 1")
0: Asynchronous interrupt on high threshold disabled for temperature sensor 1
1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1_AITLEN:** Asynchronous interrupt enable flag for low threshold on temperature sensor 1.
This bit are set and cleared by software.
It enables the asynchronous interrupt when the temperature is below the low threshold (only when REFCLK_SEL= 1)
0: Asynchronous interrupt on low threshold disabled for temperature sensor 1
1: Asynchronous interrupt on low threshold enabled for temperature sensor 1

Bit 4 **TS1_AITEEN:** Asynchronous interrupt enable flag for end of measurement on temperature sensor 1

This bit are set and cleared by software.
It enables the asynchronous interrupt for end of measurement (only when REFCLK_SEL = 1).
0: Asynchronous interrupt for end of measurement disabled on temperature sensor 1
1: Asynchronous interrupt for end of measurement enabled on temperature sensor 1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHEN:** Interrupt enable flag for high threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the interrupt when the measure reaches or is above the high threshold.
0: Synchronous interrupt for high threshold disabled on temperature sensor 1
1: Synchronous interrupt for high threshold enabled on temperature sensor 1

Bit 1 **TS1_ITLEN:** Interrupt enable flag for low threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the synchronous interrupt when the measure reaches or is below the low threshold.
0: Synchronous interrupt for low threshold disabled on temperature sensor 1
1: Synchronous interrupt for low threshold enabled on temperature sensor 1

Bit 0 **TS1_ITEEN:** Interrupt enable flag for end of measurement on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the synchronous interrupt for end of measurement.
0: Synchronous interrupt for end of measurement disabled on temperature sensor 1
1: Synchronous interrupt for end of measurement enabled on temperature sensor 1

27.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)

DTS_ICIFR is the control register for the interrupt flags.

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_ CAITHF	TS1_ CAITLF	TS1_ CAITEF	Res.	TS1_ CITHF	TS1_ CITLF	TS1_ CITEF
									rc_w1	rc_w1	rc_w1		rc_w1	rc_w1	rc_w1

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF:** Asynchronous interrupt clear flag for high threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF:** Asynchronous interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.

Bit 4 **TS1_CAITEF:** Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.
Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_CITHF:** Interrupt clear flag for high threshold on temperature sensor 1
Writing this bit to 1 clears the TS1_ITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF:** Interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF:** Interrupt clear flag for end of measurement on temperature sensor 1
Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

27.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

Reset value: 0x0000 0000

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

Bits 31:0 **TS_OP[31:0]:** general purpose option bits

27.6.10 DTS register map

The following table summarizes the temperature sensor registers.

Table 277. DTS register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**DTS_ CFGR1**	Res.	HSREF_CLK_DIV [6:0]							Res.	Res.	Q_MEAS_OPT	REFCLK_SEL	TS1_ SMP_TIME [3:0]				Res.	Res.	Res.	Res.	TS1_ INTRIG_SEL [3:0]				Res.	Res.	Res.	TS1_START	Res.	Res.	Res.	TS1_EN
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0					0	0	0	0				0				0
0x04	Reserved																																
0x08	**DTS_ T0VALR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_T0[1:0]		TS1_FMT0[15:0]															
	Reset value															X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	Reserved																																
0x10	**DTS_ RAMPVALR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RAMP_COEFF[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x14	**DTS_ITR1**	TS1_HITTHD[15:0]																TS1_LITTHD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	Reserved																																
0x1C	**DTS_DR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_MFREQ[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**DTS_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHF	TS1_AITLF	TS1_AITEF	Res.	TS1_ITHF	TS1_ITLF	TS1_ITEF
	Reset value																	0									0	0	0		0	0	0
0x24	**DTS_ITENR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_AITHEN	TS1_AITLEN	TS1_AITEEN	Res.	TS1_ITHEN	TS1_ITLEN	TS1_ITEEN
	Reset value																										0	0	0		0	0	0
0x28	**DTS_ICIFR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_CAITHF	TS1_CAITLF	TS1_CAITEF	Res.	TS1_CITHF	TS1_CITLF	TS1_CITEF
	Reset value																										0	0	0		0	0	0
0x2C	**DTS_OR**	TS_OP31	TS_OP30	TS_OP29	TS_OP28	TS_OP27	TS_OP26	TS_OP25	TS_OP24	TS_OP23	TS_OP22	TS_OP21	TS_OP20	TS_OP19	TS_OP18	TS_OP17	TS_OP16	TS_OP15	TS_OP14	TS_OP13	TS_OP12	TS_OP11	TS_OP10	TS_OP9	TS_OP8	TS_OP7	TS_OP6	TS_OP5	TS_OP4	TS_OP3	TS_OP2	TS_OP1	TS_OP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

28 Digital-to-analog converter (DAC)

28.1 DAC introduction

The DAC module is a 12-bit, voltage output digital-to-analog converter. The DAC can be configured in 8- or 12-bit mode and may be used in conjunction with the DMA controller. In 12-bit mode, the data can be left- or right-aligned. The DAC features two output channels, each with its own converter. In dual DAC channel mode, conversions can be done independently or simultaneously when both channels are grouped together for synchronous update operations. An input reference pin, VREF+ (shared with others analog peripherals) is available for better resolution. An internal reference can also be set on the same input. Refer to *voltage reference buffer (VREFBUF)* section.

The DAC output buffer can be optionally enabled to obtain a high drive output current. An individual calibration can be applied on each DAC output channel. The DAC output channels support a low power mode, the sample and hold mode.

28.2 DAC main features

The DAC main features are the following (see *Figure 269: Dual-channel DAC block diagram*)

- One DAC interface, maximum two output channels
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise-wave and Triangular-wave generation
- Dual DAC channel for independent or simultaneous conversions
- DMA capability for each channel including DMA underrun error detection
- Double data DMA capability to reduce the bus activity
- External triggers for conversion
- DAC output channel buffered/unbuffered modes
- Buffer offset calibration
- Sample and hold mode for low power operation in Stop mode
- Input voltage reference from VREF+ pin or internal VREFBUF reference

Figure 269 shows the block diagram of a DAC channel and *Table 279* gives the pin description.

28.3 DAC implementation

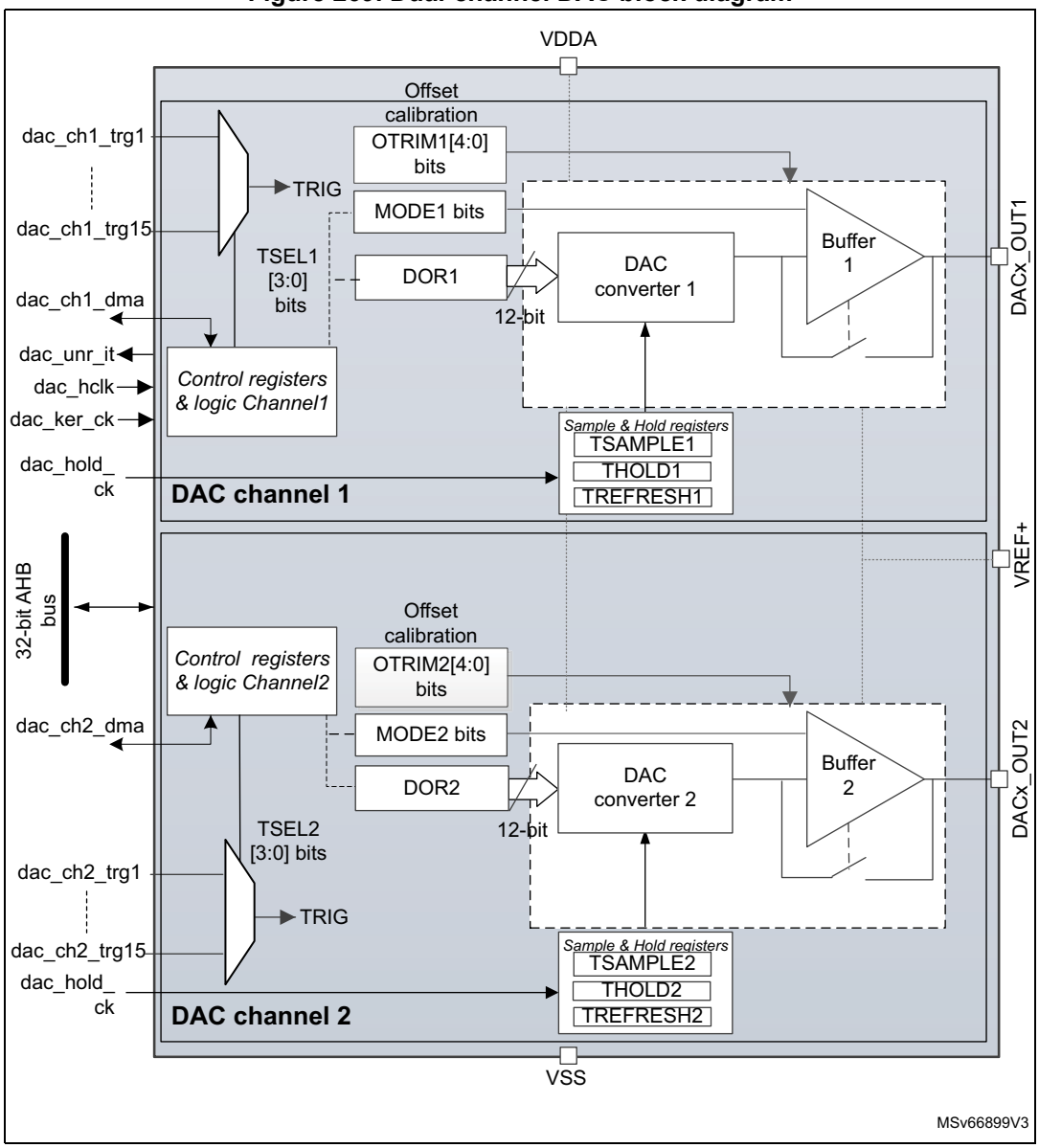
Table 278. DAC features

DAC features	DAC1
Dual channel	X
Output buffer	X
I/O connection	DAC1_OUT1 on PA4, DAC1_OUT2 on PA5
Maximum sampling time	1 Msps
Autonomous mode	-
VREF+ pin	X

28.4 DAC functional description

28.4.1 DAC block diagram

Figure 269. Dual-channel DAC block diagram



1. MODEx bits in the DAC_MCR control the output mode and allow switching between the normal mode in buffer/unbuffered configuration and the sample and hold mode.
2. Refer to *Section 28.3: DAC implementation* for channel2 availability.

28.4.2 DAC pins and internal signals

The DAC includes:

- Up to two output channels
- DAC output channel buffered or non buffered
- Sample and hold block and registers operational in Stop mode, using the LSI/LSE clock source (dac_hold_ck) for static conversion.

The DAC includes up to two separate output channels. Each output channel can be connected to on-chip peripherals such as comparator, operational amplifier and ADC (if available). In this case, the DAC output channel can be disconnected from the DACx_OUTy output pin and the corresponding GPIO can be used for another purpose.

The DAC output can be buffered or not. The sample and hold block and its associated registers can run in Stop mode using the LSI/LSE clock source (dac_hold_ck).

Table 279. DAC input/output pins

Pin name	Signal type	Remarks
VREF+	Input, analog positive reference	The higher/positive reference voltage for the DAC, $V_{REF+} \leq V_{DDAmax}$ (refer to datasheet)
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply
DACx_OUTy	Analog output signal	DACx channely analog output

Table 280. DAC internal input/output signals

Internal signal name	Signal type	Description
dac_ch1_dma	Bidirectional	DAC channel1 DMA request/acknowledge
dac_ch2_dma	Bidirectional	DAC channel2 DMA request/acknowledge
dac_ch1_trgx (x = 1 to 15)	Inputs	DAC channel1 trigger inputs
dac_ch2_trgx (x = 1 to 15)	Inputs	DAC channel2 trigger inputs
dac_unr_it	Output	DAC underrun interrupt
dac_hclk	Input	DAC peripheral clock
dac_ker_ck	Input	DAC kernel clock
dac_hold_ck	Input	DAC low-power clock used in sample and hold mode

Table 281. DAC interconnection

Signal name	Source	Source type
dac_hold_ck	ck_lsi or ck_lse	LSI or LSE clock selected in the RCC
dac_chx_trg1 (x = 1, 2)	tim1_trgo	Internal signal from on-chip timers
dac_chx_trg2 (x = 1, 2)	tim2_trgo	Internal signal from on-chip timers
dac_chx_trg3 (x = 1, 2)	tim4_trgo	Internal signal from on-chip timers
dac_chx_trg4 (x = 1, 2)	tim5_trgo	Internal signal from on-chip timers
dac_chx_trg5 (x = 1, 2)	tim6_trgo	Internal signal from on-chip timers

Table 281. DAC interconnection (continued)

Signal name	Source	Source type
dac_chx_trg6 (x = 1, 2)	tim7_trgo	Internal signal from on-chip timers
dac_chx_trg7 (x = 1, 2)	tim8_trgo	Internal signal from on-chip timers
dac_chx_trg8 (x = 1, 2)	tim15_trgo	Internal signal from on-chip timers
dac_chx_trg11 (x = 1, 2)	lptim1_ch1	Internal signal from on-chip timers
dac_chx_trg12 (x = 1, 2)	lptim2_ch1	Internal signal from on-chip timers
dac_chx_trg13 (x = 1, 2)	exti9	External pin

28.4.3 DAC clocks

Two clock sources can be used to update the DAC:

- dac_hclk: DAC peripheral clock (AHB clock)
- dac_ker_ck: DAC kernel clock: this clock can be used to synchronize DAC and ADC.
- dac_hold_ck: low-power clock used in sample and hold mode

The DAC clock is selected in the RCC.

28.4.4 DAC channel enable

Each DAC channel can be powered on by setting its corresponding ENx bit in the DAC_CR register. The DAC channel is then enabled after a t_{WAKEUP} startup time.

DACxRDY bit is set in the DAC_SR register when the DAC interface is ready to accept data. Writing new data or asserting the trigger is not allowed when ENx bit is set while DACxRDY signal is reset.

Note: The ENx bit enables the analog DAC channelx only. The DAC channelx digital interface is enabled even if the ENx bit is reset.

28.4.5 DAC data format

Depending on the selected configuration mode, the data have to be written into the specified register as described below:

- Single DAC channel

  There are three possibilities:

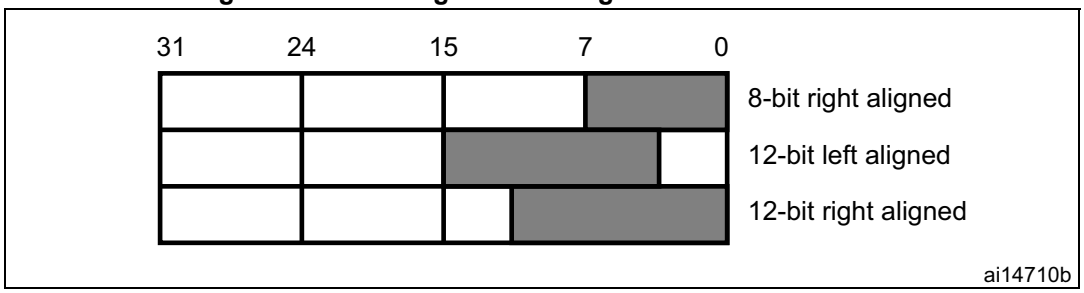
  – 8-bit right alignment: the software has to load data into the DAC_DHR8Rx[7:0] bits (stored into the DHRx[11:4] bits)

  – 12-bit left alignment: the software has to load data into the DAC_DHR12Lx [15:4] bits (stored into the DHRx[11:0] bits)

  – 12-bit right alignment: the software has to load data into the DAC_DHR12Rx [11:0] bits (stored into the DHRx[11:0] bits)

Depending on the loaded DAC_DHRyyyx register, the data written by the user is shifted and stored into the corresponding DAC_DHRx (data holding registerx, which are internal non-memory-mapped registers). The DAC_DHRx register is then loaded into the DAC_DORx register either automatically, by software trigger or by an external event trigger.

Figure 270. Data registers in single DAC channel mode



- Dual DAC channels (when available)

  There are three possibilities:

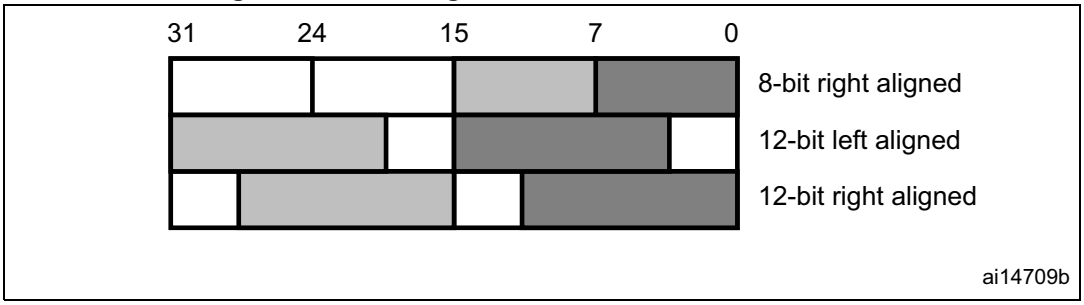
  – 8-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR8RD [7:0] bits (stored into the DHR1[11:4] bits) and data for DAC channel2 to be loaded into the DAC_DHR8RD [15:8] bits (stored into the DHR2[11:4] bits)

  – 12-bit left alignment: data for DAC channel1 to be loaded into the DAC_DHR12LD [15:4] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12LD [31:20] bits (stored into the DHR2[11:0] bits)

  – 12-bit right alignment: data for DAC channel1 to be loaded into the DAC_DHR12RD [11:0] bits (stored into the DHR1[11:0] bits) and data for DAC channel2 to be loaded into the DAC_DHR12RD [27:16] bits (stored into the DHR2[11:0] bits)

Depending on the loaded DAC_DHRyyyD register, the data written by the user is shifted and stored into DHR1 and DHR2 (data holding registers, which are internal non-memory-mapped registers). The DHR1 and DHR2 registers are then loaded into the DAC_DOR1 and DOR2 registers, respectively, either automatically, by software trigger or by an external event trigger.

Figure 271. Data registers in dual DAC channel mode



Signed/unsigned data

DAC input data are unsigned: 0x000 corresponds to the minimum value and 0xFFF to the maximum value for 12-bit mode.

The DAC can also handle signed input data in 2’s complement format. This is done by setting SINFORMATx bit in the DAC_MCR register.

When SINFORMATx bit is set, the MSB of the data written to DAC_DHRx registers is inverted when it is copied to the DAC_DORx register, and the DAC interface can accept signed data (Q1.15, Q1.11 or Q1.7 format). DAC_DHR12Lx register can be used to store 16-bit signed data in the data holding registers. The 12 MSBs of 16-bit data are used for the DAC output data and the MSB is inverted. The four LSBs are simply ignored.

Table 282. Data format (case of 12-bit data)

SINFORMATx bit	DATA written to DAC_DHRx register	DATA transfered to DAC_DORx register
0	0x000	0x000
0	0xFFF	0xFFF
1	0x7FF	0xFFF
1	0x000	0x800
1	0xFFF	0x7FF
1	0x800	0x000

28.4.6 DAC conversion

The DAC_DORx cannot be written directly and any data transfer to the DAC channelx must be performed by loading the DAC_DHRx register (write operation to DAC_DHR8Rx, DAC_DHR12Lx, DAC_DHR12Rx, DAC_DHR8RD, DAC_DHR12RD or DAC_DHR12LD).

Data stored in the DAC_DHRx register are automatically transferred to the DAC_DORx register after one dac_hclk clock cycle, if no hardware trigger is selected (TENx bit in DAC_CR register is reset). However, when a hardware trigger is selected (TENx bit in DAC_CR register is set) and a trigger occurs, the transfer is performed three dac_hclk clock cycles after the trigger signal.

When DAC_DORx is loaded with the DAC_DHRx contents, the analog output voltage becomes available after a time $t_{SETTLING}$ that depends on the power supply voltage and the analog output load.

To synchronize DAC and ADC, the same clock source can be used for both peripherals. This is done by selecting the dac_ker_ck clock instead of the dac_hclk clock (AHB clock) in the RCC.

HFSEL bits of DAC_MCR must be set when dac_ker_ck clock speed is faster than 80 MHz.

Refer to Table *HFSEL description* below for the limitation of the DAC_DORx update rate depending on HFSEL bits and dac_hclk or dac_ker_ck clock frequency.

If the data is updated or a software/hardware trigger event occurs during the non-allowed period, the peripheral behavior is unpredictable.

The above timing is only related to the limitation of the DAC interface. Refer also to the $t_{SETTLING}$ parameter value in the product datasheet.

Table 283. HFSEL description

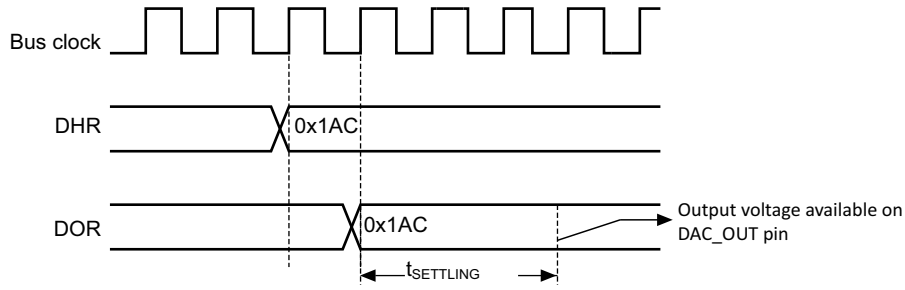
HFSEL [1:0]	Clock frequency	Latency using AHB clock (dac_hclk)	Latency using dac_ker_ck clock	Function
00	< 80 MHz	3	4	DAC_DOR update rate up to 3 AHB clock cycles or 4 dac_ker_ck cycles.
01	≥ 80 MHz(1)	5	5	DAC_DOR update rate up to 5 AHB clock or dac_ker_ck cycles.

Table 283. HFSEL description (continued)

HFSEL [1:0]	Clock frequency	Latency using AHB clock (dac_hclk)	Latency using dac_ker_ck clock	Function
10	≥ 160 MHz	7	6	DAC_DOR update rate up to 7 AHB clock cycles or 6 dac_ker_ck cycles.
11	Reserved	-	-	-

1. Refer to the device datasheet for the value of the maximum dac_hclk or dac_ker_ck frequency.

Figure 272. Timing diagram for conversion with trigger disabled TEN = 0



28.4.7 DAC output voltage

Digital inputs are converted to output voltages on a linear conversion between 0 and V_{REF+}.

The analog output voltages on each DAC channel pin are determined by the following equation:

$$\text{DAC output} = V_{REF} \times \frac{DOR}{4096}$$

where all voltages are expressed in Volt.

28.4.8 DAC trigger selection

If the TENx control bit is set, the conversion can then be triggered by an external event (timer counter, external interrupt line). The TSELx[3:0] control bits determine which out of 16 possible events triggers the conversion as shown in TSELx[3:0] bits of the DAC_CR register. These events can be either the software trigger or hardware triggers. Refer to the interconnection table in *Section 28.4.2*.

Each time a DAC interface detects a rising edge on the selected trigger source (refer to the table below), the last data stored into the DAC_DHRx register are transferred into the DAC_DORx register. The DAC_DORx register is updated three dac_hclk cycles after the trigger occurs.

If the software trigger is selected, the conversion starts once the SWTRIG bit is set. SWTRIG is reset by hardware once the DAC_DORx register has been loaded with the DAC_DHRx register contents.

Note: TSELx[3:0] bit cannot be changed when the ENx bit is set.

When software trigger is selected, the transfer from the DAC_DHRx register to the DAC_DORx register takes only one dac_hclk clock cycle.

28.4.9 DMA requests

Each DAC channel has a DMA capability. Two DMA channels are used to service DAC channel DMA requests.

When an external trigger (but not a software trigger) occurs while the DMAENx bit is set, the value of the DAC_DHRx register is transferred into the DAC_DORx register when the transfer is complete, and a DMA request is generated.

In dual mode, if both DMAENx bits are set, two DMA requests are generated. If only one DMA request is needed, only the corresponding DMAENx bit must be set. In this way, the application can manage both DAC channels in dual mode by using one DMA request and a unique DMA channel.

As DAC_DHRx to DAC_DORx data transfer occurred before the DMA request, the very first data has to be written to the DAC_DHRx before the first trigger event occurs.

DMA underrun

The DAC DMA request is not queued so that if a second external trigger arrives before the acknowledgment for the first external trigger is received (first request), then no new request is issued and the DMA channelx underrun flag DMAUDRx in the DAC_SR register is set, reporting the error condition. The DAC channelx continues to convert old data.

The software must clear the DMAUDRx flag by writing 1, clear the DMAEN bit of the used DMA stream and re-initialize both DMA and DAC channelx to restart the transfer correctly. The software must modify the DAC trigger conversion frequency or lighten the DMA workload to avoid a new DMA underrun. Finally, the DAC conversion can be resumed by enabling both DMA data transfer and conversion trigger.

For each DAC channelx, an interrupt is also generated if its corresponding DMAUDRIEx bit in the DAC_CR register is enabled.

DMA double data mode

When the DMA controller is used in normal mode, only 12-bit (or 8-bit) data are transferred by a DMA request. As the AHB width is 32 bits, two 12-bit data may be transferred simultaneously. To use this mode, set the DMADOUBLEx bit of DAC_MCR register.

A DAC DMA request is generated every two external triggers (except for software triggers) when the DMAENx bit is set:

1. When the first trigger is detected, the value of the DAC_DHRx and DAC_DHRBx registers are transferred into the DAC_DORx and DAC_DORBx registers. The actual DAC data is loaded into the DAC_DORx register. A DMA request is then generated. The DMA writes the new data to the DAC_DHRx and DAC_DHRBx data registers.
2. When the next trigger is detected, the actual DAC data is loaded into the DAC_DHRBx register. This second trigger does not generate any DMA request. The DORSTATx bit indicates which DOR data is actually loaded into the analog DAC input.

DMA underrun function is also supported in DMA double data mode.

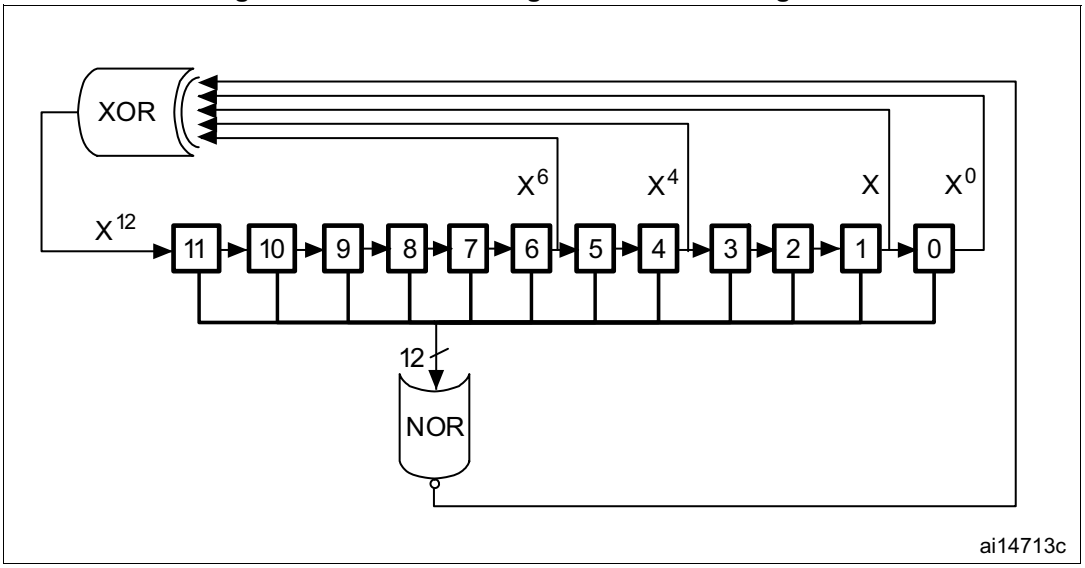
The following conditions must be met to change from double data to single data mode or vice versa:

- The DAC must be disabled.
- DMAEN bit must be cleared (ENx = 0 and DMAEN = 0).

28.4.10 Noise generation

In order to generate a variable-amplitude pseudonoise, an LFSR (linear feedback shift register) is available. DAC noise generation is selected by setting WAVEx[1:0] to 01. The preloaded value in LFSR is 0xAAA. This register is updated three dac_hclk clock cycles after each trigger event, following a specific calculation algorithm.

Figure 273. DAC LFSR register calculation algorithm

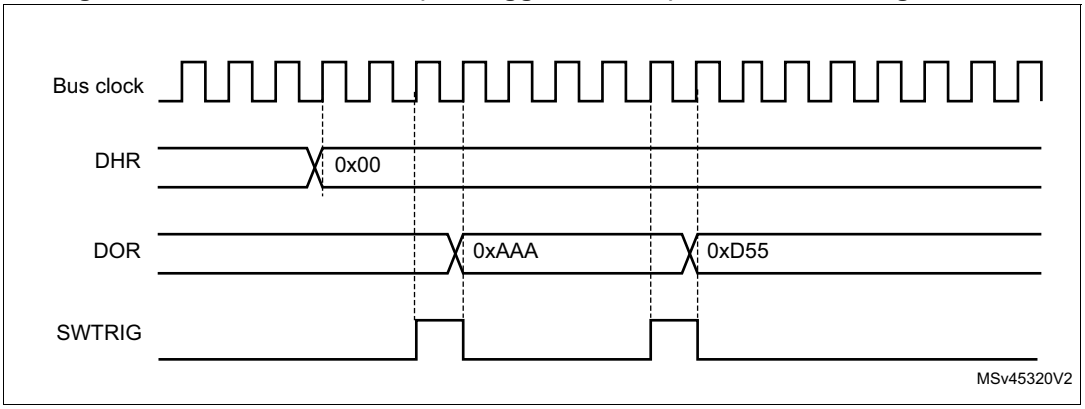


The LFSR value, that may be masked partially or totally by means of the MAMPx[3:0] bits in the DAC_CR register, is added up to the DAC_DHRx contents without overflow and this value is then transferred into the DAC_DORx register.

If LFSR is 0x0000, a 1 is injected into it (antilock-up mechanism).

It is possible to reset LFSR wave generation by resetting the WAVEx[1:0] bits.

Figure 274. DAC conversion (SW trigger enabled) with LFSR wave generation



*Note: The DAC trigger must be enabled for noise generation by setting the TEN*x *bit in the DAC_CR register.*

28.4.11 Triangle-wave generation

It is possible to add a small-amplitude triangular waveform on a DC or slowly varying signal. DAC triangle-wave generation is selected by setting WAVEx[1:0] to 10. The amplitude is configured through the MAMPx[3:0] bits in the DAC_CR register. An internal triangle counter is incremented three dac_hclk clock cycles after each trigger event. The value of this counter is then added to the DAC_DHRx register without overflow and the sum is transferred into the DAC_DORx register. The triangle counter is incremented as long as it is less than the maximum amplitude defined by the MAMPx[3:0] bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again and so on.

It is possible to reset triangle wave generation by resetting the WAVEx[1:0] bits.

Figure 275. DAC triangle wave generation

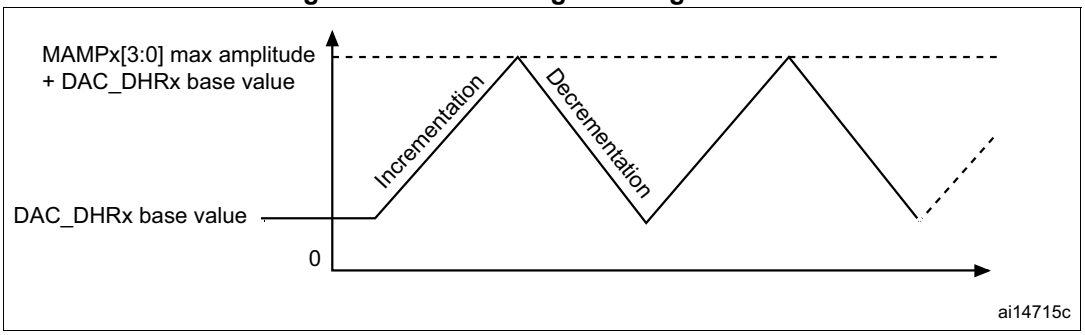
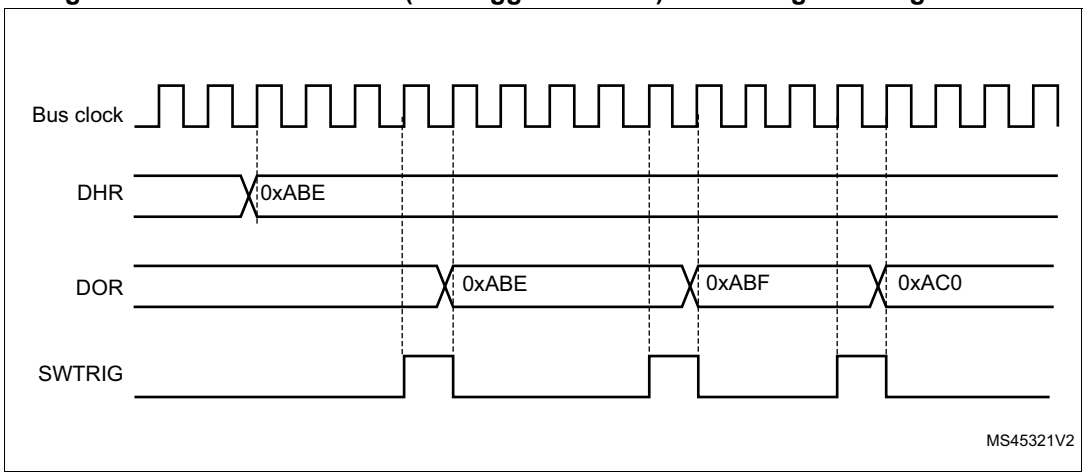


Figure 276. DAC conversion (SW trigger enabled) with triangle wave generation



Note: The DAC trigger must be enabled for triangle wave generation by setting the TENx bit in the DAC_CR register.

The MAMPx[3:0] bits must be configured before enabling the DAC, otherwise they cannot be changed.

28.4.12 DAC channel modes

Each DAC channel can be configured in normal mode or sample and hold mode. The output buffer can be enabled to obtain a high drive capability. Before enabling output buffer, the voltage offset needs to be calibrated. This calibration is performed at the factory (loaded after reset) and can be adjusted by software during application operation.

Normal mode

In normal mode, there are four combinations, by changing the buffer state and by changing the DACx_OUTy pin interconnections.

To enable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 000: DAC is connected to the external pin

To disable the output buffer, the MODEx[2:0] bits in DAC_MCR register must be:

- 010: DAC is connected to the external pin

Sample and hold mode

In sample and hold mode, the DAC core converts data on a triggered conversion, and then holds the converted voltage on a capacitor. When not converting, the DAC cores and buffer are completely turned off between samples and the DAC output is tri-stated, therefore reducing the overall power consumption. A stabilization period, which value depends on the buffer state, is required before each new conversion.

In this mode, the DAC core and all corresponding logic and registers are driven by the LSI/LSE low-speed clock (dac_hold_ck) in addition to the dac_hclk clock, allowing using the DAC channels in deep low power modes such as Stop mode.

The LSI/LSE low-speed clock (dac_hold_ck) must not be stopped when the sample and hold mode is enabled.

The sample/hold mode operations can be divided into three phases:

1. Sample phase: the sample/hold element is charged to the desired voltage. The charging time depends on capacitor value (internal or external, selected by the user). The sampling time is configured with the TSAMPLEx[9:0] bits in DAC_SHSRx register. During the write of the TSAMPLEx[9:0] bits, the BWSTx bit in DAC_SR register is set to 1 to synchronize between both clocks domains (AHB and low speed clock) and allowing the software to change the value of sample phase during the DAC channel operation
2. Hold phase: the DAC output channel is tri-stated, the DAC core and the buffer are turned off, to reduce the current consumption. The hold time is configured with the THOLDx[9:0] bits in DAC_SHHR register
3. Refresh phase: the refresh time is configured with the TREFRESHx[7:0] bits in DAC_SHRR register

The timings for the three phases above are in units of LSI/LSE clock periods. As an example, to configure a sample time of 350 µs, a hold time of 2 ms and a refresh time of 100 µs assuming LSI/LSE ~32 KHz is selected:

- 12 cycles are required for sample phase: TSAMPLEx[9:0] = 11.
- 62 cycles are required for hold phase: THOLDx[9:0] = 62.
- and 4 cycles are required for refresh period: TREFRESHx[7:0] = 4.

In this example, the power consumption is reduced by almost a factor of 15 versus normal modes.

The formulas to compute the right sample and refresh timings are described in the table below, the Hold time depends on the leakage current.

Table 284. Sample and refresh timings

Buffer State	t_{SAMP} (1)(2)	$t_{REFRESH}$ (2)(3)
Enable	7 µs + (10*R_{BON}*C_{SH})	7 µs + (R_{BON}*C_{SH})*ln(2*N_{LSB})
Disable	3 µs + (10*R_{BOFF}*C_{SH})	3 µs + (R_{BOFF}*C_{SH})*ln(2*N_{LSB})

1. In the above formula, the settling to the desired code value with ½ LSB or accuracy requires 10 constant time for 12 bits resolution. For 8-bit resolution, the settling time is 7 constant time.
2. C_{SH} is the capacitor in sample and hold mode.
3. The tolerated voltage drop during the hold phase “Vd” is represented by the number of LSBs after the capacitor discharging with the output leakage current. The settling back to the desired value with ½ LSB error accuracy requires ln(2*Nlsb) constant time of the DAC.

Example of the sample and refresh time calculation with output buffer on

The values used in the example below are provided as indication only. Refer to the product datasheet for product data.

C_{SH} = 100 nF

V_{REF+} = 3.0 V

Sampling phase:

t_{SAMP} = 7 µs + (10 * 2000 * 100 * 10^{-9}) = 2.007 ms

(where R_{BON} = 2 kΩ)

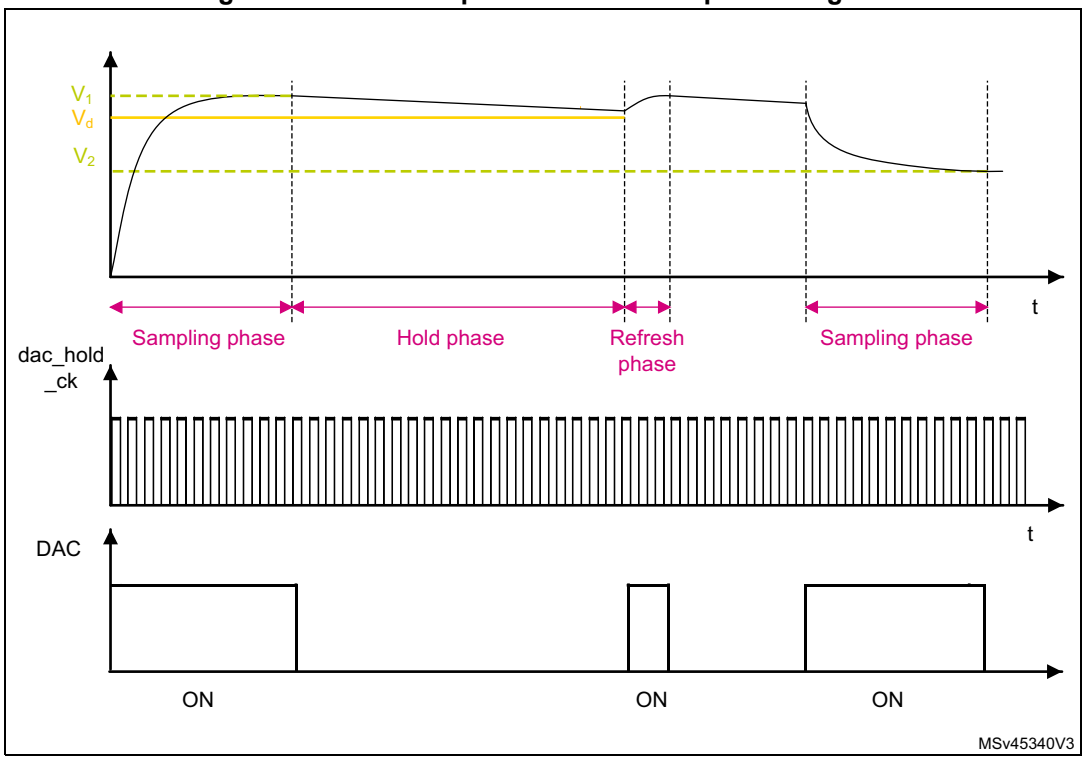
Refresh phase:

$t_{REFRESH}$ = 7 µs + (2000 * 100 * 10^{-9}) * ln(2*10) = 606.1 µs
(where N_{LSB} = 10 (10 LSB drop during the hold phase)

Hold phase:

$D_V = i_{leak} * t_{hold} / C_{SH}$ = 0.0073 V (10 LSB of 12bit at 3 V)
i_{leak} = 150 nA (worst case on the IO leakage on all the temperature range)
t_{hold} = 0.0073 * 100 * 10^{-9} / (150 * 10^{-9}) = 4.867 ms

Figure 277. DAC sample and hold mode phase diagram



Like in normal mode, the sample and hold mode has different configurations.

To enable the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 100: DAC is connected to the external pin

To disabled the output buffer, MODEx[2:0] bits in DAC_MCR register must be set to:

- 110: DAC is connected to external pin

  When MODEx[2:0] bits are equal to 111, an internal capacitor, C_{Lint}, holds the voltage output of the DAC core and then drive it to on-chip peripherals.

All sample and hold phases are interruptible, and any change in DAC_DHRx immediately triggers a new sample phase.

Table 285. Channel output modes summary

MODEx[2:0]			Mode	Buffer	Output connections
0	0	0	Normal mode	Enabled	Connected to external pin
0	1	0		Disabled	Connected to external pin
1	0	0	Sample and hold mode	Enabled	Connected to external pin
1	1	0		Disabled	Connected to external pin and to on chip peripherals (such as comparators)

28.4.13 DAC channel buffer calibration

The transfer function for an N-bit digital-to-analog converter (DAC) is:

$$V_{out} = \left((D / 2^N) \times G \times V_{REF}\right) + V_{OS}$$

Where V_{OUT} is the analog output, D is the digital input, G is the gain, V_{REF} is the nominal full-scale voltage, and V_{OS} is the offset voltage. For an ideal DAC channel, G = 1 and V_{OS} = 0.

Due to output buffer characteristics, the voltage offset may differ from part-to-part and introduce an absolute offset error on the analog output. To compensate the V_{OS}, a calibration is required by a trimming technique.

The calibration is only valid when the DAC channelx is operating with buffer enabled (MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101). if applied in other modes when the buffer is off, it has no effect. During the calibration:

- The buffer output is disconnected from the pin internal/external connections and put in tristate mode (HiZ).
- The buffer acts as a comparator to sense the middle-code value 0x800 and compare it to V_{REF+}/2 signal through an internal bridge, then toggle its output signal to 0 or 1 depending on the comparison result (CAL_FLAGx bit).

Two calibration techniques are provided:

- Factory trimming (default setting)

  The DAC buffer offset is factory trimmed. The default value of OTRIMx[4:0] bits in DAC_CCR register is the factory trimming value and it is loaded once DAC digital interface is reset.
- User trimming

  The user trimming can be done when the operating conditions differs from nominal factory trimming conditions and in particular when V_{DDA} voltage, temperature, V_{REF+} values change and can be done at any point during application by software.

Note: Refer to the datasheet for more details of the nominal factory trimming conditions.

In addition, when V_{DD} is removed (example the device enters in Standby or VBAT modes) the calibration is required.

The steps to perform a user trimming calibration are as below:

1. If the DAC channel is active, write 0 to ENx bit in DAC_CR to disable the channel.
2. Select a mode where the buffer is enabled, by writing to DAC_MCR register, MODEx[2:0] = 0b000 or 0b001 or 0b100 or 0b101.
3. Start the DAC channelx calibration, by setting the CENx bit in DAC_CR register to 1.
4. Apply a trimming algorithm:
   a) Write a code into OTRIMx[4:0] bits, starting by 0b00000.
   b) Wait for t_{TRIM} delay.
   c) Check if CAL_FLAGx bit in DAC_SR is set to 1.
   d) Until the CAL_FLAGx is read as 1 or the maximum trimming code is reached, increment OTRIMx[4:0] and repeat substeps from (b) to (d).

The software algorithm may use either a successive approximation or dichotomy techniques to compute and set the content of OTRIMx[4:0] bits in a faster way.

Note: A t_{TRIM} delay must be respected between the write to the OTRIMx[4:0] bits and the read of the CAL_FLAGx bit in DAC_SR register in order to get a correct value.This parameter is specified into datasheet electrical characteristics section.

If V_{DDA}, V_{REF+} and temperature conditions do not change during device operation while it enters more often in Standby and VBAT modes, the software may store the OTRIMx[4:0] bits found in the first user calibration in the flash or in back-up registers. then to load/write them directly when the device power is back again thus avoiding to wait for a new calibration time.

When CENx bit is set, it is not allowed to set ENx bit.

28.4.14 DAC channel conversion modes

Four conversion modes are possible.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Load the DAC channel data into the desired DHR registers (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DHRx register is transferred into DAC_DORx (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bit, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the DAC channel trigger enable bits, TENx.
2. Configure the trigger sources by setting different values in the TSELx[3:0] bits.
3. Configure the DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the DAC channel data into the desired DHR register (DAC_DHR12R1, DAC_DHR12L1 or DAC_DHR8R1).

When a DAC channel trigger arrives, the DAC channel triangle counter, with the same triangle amplitude, is added to the DHRx register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel triangle counter is then updated.

Independent trigger with single sawtooth generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Configure the trigger sources by setting different values in STRSTTRIGSELx[3:0] and STINCTRIGSELx[3:0] bits.
2. Configure the DAC channel WAVEx[1:0] bits to 11 and set the same STRSTDATAx[11:0], STINCDATAx[15:0] and STDIRx values for each register.

When a DAC channel trigger arrives, the DAC channel sawtooth counter updates the DHRx register and transfers it into DAC_DOR1 (three AHB clock cycles later).

28.4.15 Dual DAC channel conversion modes (if dual channels are available)

To efficiently use the bus bandwidth in applications that require the two DAC channels at the same time, three dual registers are implemented: DHR8RD, DHR12RD and DHR12LD. A unique register access is then required to drive both DAC channels at the same time. For the wave generation, no accesses to DHRxxxD registers are required. As a result, two output channels can be used either independently or simultaneously.

15 conversion modes are possible using the two DAC channels and these dual registers. All the conversion modes can nevertheless be obtained using separate DAC_DHRx registers if needed.

All modes are described in the paragraphs below.

Independent trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DHR1 register is transferred into DAC_DOR1 (three dac_hclk clock cycles later).

When a DAC channel2 trigger arrives, the DHR2 register is transferred into DAC_DOR2 (three dac_hclk clock cycles later).

Independent trigger with single LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR masks values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). Then the LFSR1 counter is updated.

When a DAC channel2 trigger arrives, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the LFSR2 counter is updated.

Independent trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value in the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Independent trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure different trigger sources by setting different values in the TSEL1 and TSEL2 bits.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a DAC channel1 trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

When a DAC channel2 trigger arrives, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous software start

To configure the DAC in this conversion mode, the following sequence is required:

- Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

In this configuration, one dac_hclk clock cycle later, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively.

Simultaneous trigger without wave generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2.
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Load the dual DAC channel data to the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DHR1 and DHR2 registers are transferred into DAC_DOR1 and DAC_DOR2, respectively (after three dac_hclk clock cycles).

Simultaneous trigger with single LFSR generation

1. To configure the DAC in this conversion mode, the following sequence is required:
2. Set the two DAC channel trigger enable bits TEN1 and TEN2.
3. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
4. Configure the two DAC channel WAVEx[1:0] bits as 01 and the same LFSR mask value in the MAMPx[3:0] bits.
5. Load the dual DAC channel data to the desired DHR register (DHR12RD, DHR12LD or DHR8RD).

When a trigger arrives, the LFSR1 counter, with the same mask, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated. At the same time, the LFSR2 counter, with the same mask, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with different LFSR generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 01 and set different LFSR mask values using the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the LFSR1 counter, with the mask configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The LFSR1 counter is then updated.
At the same time, the LFSR2 counter, with the mask configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The LFSR2 counter is then updated.

Simultaneous trigger with single triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and the same maximum amplitude value using the MAMPx[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with the same triangle amplitude, is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three dac_hclk clock cycles later). The DAC channel1 triangle counter is then updated.

At the same time, the DAC channel2 triangle counter, with the same triangle amplitude, is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). The DAC channel2 triangle counter is then updated.

Simultaneous trigger with different triangle generation

To configure the DAC in this conversion mode, the following sequence is required:

1. Set the two DAC channel trigger enable bits TEN1 and TEN2
2. Configure the same trigger source for both DAC channels by setting the same value in the TSEL1 and TSEL2 bitfields.
3. Configure the two DAC channel WAVEx[1:0] bits as 1x and set different maximum amplitude values in the MAMP1[3:0] and MAMP2[3:0] bits.
4. Load the dual DAC channel data into the desired DHR register (DAC_DHR12RD, DAC_DHR12LD or DAC_DHR8RD).

When a trigger arrives, the DAC channel1 triangle counter, with a triangle amplitude configured by MAMP1[3:0], is added to the DHR1 register and the sum is transferred into DAC_DOR1 (three AHB clock cycles later). Then the DAC channel1 triangle counter is updated.

At the same time, the DAC channel2 triangle counter, with a triangle amplitude configured by MAMP2[3:0], is added to the DHR2 register and the sum is transferred into DAC_DOR2 (three dac_hclk clock cycles later). Then the DAC channel2 triangle counter is updated.

28.5 DAC in low-power modes

Table 286. Effect of low-power modes on DAC

Mode	Description
Sleep	No effect, DAC used with DMA.
Stop(1)	The DAC remains active with a static value. The sample and hold mode can be selected using LSE/LSI clock.
Standby	The DAC peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 28.3: DAC implementation* for information on the Stop modes supported by the DAC peripheral.

28.6 DAC interrupts

Table 287. DAC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop mode	Exit Standby mode
DAC	DMA underrun	DMAUDRx	DMAUDRIEx	Write DMAUDRx = 1	Yes	No	No

28.7 DAC registers

Refer to *Section 1 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32-bit).

28.7.1 DAC control register (DAC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[3]	TSEL2[2]	TSEL2[1]	TSEL2[0]	TEN2	EN2
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3]	TSEL1[2]	TSEL1[1]	TSEL1[0]	TEN1	EN1
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CEN2**: DAC channel2 calibration enable

This bit is set and cleared by software to enable/disable DAC channel2 calibration, it can be written only if EN2 bit is set to 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel2 in normal operating mode

1: DAC channel2 in calibration mode

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 29 **DMAUDRIE2**: DAC channel2 DMA underrun interrupt enable

This bit is set and cleared by software.

0: DAC channel2 DMA underrun interrupt disabled

1: DAC channel2 DMA underrun interrupt enabled

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 28 **DMAEN2**: DAC channel2 DMA enable
This bit is set and cleared by software.
0: DAC channel2 DMA mode disabled
1: DAC channel2 DMA mode enabled
Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 27:24 **MAMP2[3:0]**: DAC channel2 mask/amplitude selector
These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.
0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1
0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3
0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7
0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15
0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31
0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63
0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127
0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255
1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511
1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023
1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047
≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095
Note: These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 23:22 **WAVE2[1:0]**: DAC channel2 noise/triangle wave generation enable
These bits are set/reset by software.
00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Reserved
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled)
These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 21:18 **TSEL2[3:0]**: DAC channel2 trigger selection
These bits select the external event used to trigger DAC channel2
0000: SWTRIG2
0001: dac_ch2_trg1
0010: dac_ch2_trg2
...
1111: dac_ch2_trg15
Refer to the trigger selection tables in *Section 28.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
Note: Only used if bit TEN2 = 1 (DAC channel2 trigger enabled).
These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 17 **TEN2**: DAC channel2 trigger enable

This bit is set and cleared by software to enable/disable DAC channel2 trigger

0: DAC channel2 trigger disabled and data written into the DAC_DHR2 register are transferred one dac_hclk clock cycle later to the DAC_DOR2 register

1: DAC channel2 trigger enabled and data from the DAC_DHR2 register are transferred three dac_hclk clock cycles later to the DAC_DOR2 register

Note: When software trigger is selected, the transfer from the DAC_DHR2 register to the DAC_DOR2 register takes only one dac_hclk clock cycle.

These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 16 **EN2**: DAC channel2 enable

This bit is set and cleared by software to enable/disable DAC channel2.

0: DAC channel2 disabled

1: DAC channel2 enabled

Note: These bits are available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CEN1**: DAC channel1 calibration enable

This bit is set and cleared by software to enable/disable DAC channel1 calibration, it can be written only if bit EN1 = 0 into DAC_CR (the calibration mode can be entered/exit only when the DAC channel is disabled) Otherwise, the write operation is ignored.

0: DAC channel1 in normal operating mode

1: DAC channel1 in calibration mode

Bit 13 **DMAUDRIE1**: DAC channel1 DMA Underrun Interrupt enable

This bit is set and cleared by software.

0: DAC channel1 DMA Underrun Interrupt disabled

1: DAC channel1 DMA Underrun Interrupt enabled

Bit 12 **DMAEN1**: DAC channel1 DMA enable

This bit is set and cleared by software.

0: DAC channel1 DMA mode disabled

1: DAC channel1 DMA mode enabled

Bits 11:8 **MAMP1[3:0]**: DAC channel1 mask/amplitude selector

These bits are written by software to select mask in wave generation mode or amplitude in triangle generation mode.

0000: Unmask bit0 of LFSR/ triangle amplitude equal to 1

0001: Unmask bits[1:0] of LFSR/ triangle amplitude equal to 3

0010: Unmask bits[2:0] of LFSR/ triangle amplitude equal to 7

0011: Unmask bits[3:0] of LFSR/ triangle amplitude equal to 15

0100: Unmask bits[4:0] of LFSR/ triangle amplitude equal to 31

0101: Unmask bits[5:0] of LFSR/ triangle amplitude equal to 63

0110: Unmask bits[6:0] of LFSR/ triangle amplitude equal to 127

0111: Unmask bits[7:0] of LFSR/ triangle amplitude equal to 255

1000: Unmask bits[8:0] of LFSR/ triangle amplitude equal to 511

1001: Unmask bits[9:0] of LFSR/ triangle amplitude equal to 1023

1010: Unmask bits[10:0] of LFSR/ triangle amplitude equal to 2047

≥ 1011: Unmask bits[11:0] of LFSR/ triangle amplitude equal to 4095

Bits 7:6 **WAVE1[1:0]**: DAC channel1 noise/triangle wave generation enable
These bits are set and cleared by software.
00: wave generation disabled
01: Noise wave generation enabled
10: Triangle wave generation enabled
11: Reserved
Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bits 5:2 **TSEL1[3:0]**: DAC channel1 trigger selection
These bits select the external event used to trigger DAC channel1
0000: SWTRIG1
0001: dac_ch1_trg1
0010: dac_ch1_trg2
...
1111: dac_ch1_trg15
Refer to the trigger selection tables in *Section 28.4.2: DAC pins and internal signals* for details on trigger configuration and mapping.
Note: Only used if bit TEN1 = 1 (DAC channel1 trigger enabled).

Bit 1 **TEN1**: DAC channel1 trigger enable
This bit is set and cleared by software to enable/disable DAC channel1 trigger.
0: DAC channel1 trigger disabled and data written into the DAC_DHR1 register are transferred one dac_hclk clock cycle later to the DAC_DOR1 register
1: DAC channel1 trigger enabled and data from the DAC_DHR1 register are transferred three dac_hclk clock cycles later to the DAC_DOR1 register
Note: When software trigger is selected, the transfer from the DAC_DHR1 *register to the DAC_DOR1 register takes only one dac_hclk clock cycle.*

Bit 0 **EN1**: DAC channel1 enable
This bit is set and cleared by software to enable/disable DAC channel1.
0: DAC channel1 disabled
1: DAC channel1 enabled

28.7.2 DAC software trigger register (DAC_SWTRGR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
														w	w

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

Bit 1 **SWTRIG2**: DAC channel2 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR2 register value has been loaded into the DAC_DOR2 register.

This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 0 **SWTRIG1**: DAC channel1 software trigger

This bit is set by software to trigger the DAC in software trigger mode.

0: No trigger

1: Trigger

Note: This bit is cleared by hardware (one dac_hclk clock cycle later) once the DAC_DHR1 register value has been loaded into the DAC_DOR1 register.

28.7.3 DAC channel1 12-bit right-aligned data holding register (DAC_DHR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DHRB[11:0]**: DAC channel1 12-bit right-aligned data B

These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

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

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data

These bits are written by software. They specify 12-bit data for DAC channel1.

28.7.4 DAC channel1 12-bit left aligned data holding register (DAC_DHR12L1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC1DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC1DHRB[11:0]**: DAC channel1 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software.
They specify 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

28.7.5 DAC channel1 8-bit right aligned data holding register (DAC_DHR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHRB[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC1DHRB[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1 when the DAC operates in double data mode.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel1.

28.7.6 DAC channel2 12-bit right aligned data holding register (DAC_DHR12R2)

This register is available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHRB[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHRB[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in DMA double data mode.

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

Bits 11:0 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software. They specify 12-bit data for DAC channel2.

28.7.7 DAC channel2 12-bit left aligned data holding register (DAC_DHR12L2)

This register is available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHRB[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHRB[11:0]**: DAC channel2 12-bit left-aligned data B
These bits are written by software. They specify 12-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specify 12-bit data for DAC channel2.

Bits 3:0 Reserved, must be kept at reset value.

28.7.8 DAC channel2 8-bit right-aligned data holding register (DAC_DHR8R2)

This register is available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHRB[7:0]								DACC2DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHRB[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software. They specify 8-bit data for DAC channel2 when the DAC operates in double data mode.

Bits 7:0 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

28.7.9 Dual DAC 12-bit right-aligned data holding register (DAC_DHR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DHR[11:0]**: DAC channel2 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

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

Bits 11:0 **DACC1DHR[11:0]**: DAC channel1 12-bit right-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

28.7.10 Dual DAC 12-bit left aligned data holding register (DAC_DHR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DACC2DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC1DHR[11:0]												Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bits 31:20 **DACC2DHR[11:0]**: DAC channel2 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel2.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:4 **DACC1DHR[11:0]**: DAC channel1 12-bit left-aligned data
These bits are written by software which specifies 12-bit data for DAC channel1.

Bits 3:0 Reserved, must be kept at reset value.

28.7.11 Dual DAC 8-bit right aligned data holding register (DAC_DHR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DACC2DHR[7:0]								DACC1DHR[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DACC2DHR[7:0]**: DAC channel2 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel2.

Bits 7:0 **DACC1DHR[7:0]**: DAC channel1 8-bit right-aligned data
These bits are written by software which specifies 8-bit data for DAC channel1.

28.7.12 DAC channel1 data output register (DAC_DOR1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC1DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC1DORB[11:0]**: DAC channel1 data output
These bits are read-only. They contain data output for DAC channel1 B.

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

Bits 11:0 **DACC1DOR[11:0]**: DAC channel1 data output
These bits are read-only, they contain data output for DAC channel1.

28.7.13 DAC channel2 data output register (DAC_DOR2)

This register is available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	DACC2DORB[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DACC2DORB[11:0]**: DAC channel2 data output
These bits are read-only. They contain data output for DAC channel2 B.

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

Bits 11:0 **DACC2DOR[11:0]**: DAC channel2 data output
These bits are read-only, they contain data output for DAC channel2.

28.7.14 DAC status register (DAC_SR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	rc_w1	r	r											

Bit 31 **BWST2**: DAC channel2 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable. It is set each time the software writes the register DAC_SHSR2, It is cleared by hardware when the write operation of DAC_SHSR2 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR2 ongoing: DAC_SHSR2 can be written

1:There is a write operation of DAC_SHSR2 ongoing: DAC_SHSR2 cannot be written

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 30 **CAL_FLAG2**: DAC channel2 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 29 **DMAUDR2**: DAC channel2 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel2

1: DMA underrun error condition occurred for DAC channel2 (the currently selected trigger is driving DAC channel2 conversion at a frequency higher than the DMA service capability rate).

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 28 **DORSTAT2**: DAC channel2 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 27 **DAC2RDY**: DAC channel2 ready status bit

This bit is set and cleared by hardware.

0: DAC channel2 is not yet ready to accept the trigger nor output data

1: DAC channel2 is ready to accept the trigger or output data

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 26:16 Reserved, must be kept at reset value.

Bit 15 **BWST1**: DAC channel1 busy writing sample time flag

This bit is systematically set just after sample and hold mode enable and is set each time the software writes the register DAC_SHSR1, It is cleared by hardware when the write operation of DAC_SHSR1 is complete. (It takes about 3 LSI/LSE periods of synchronization).

0:There is no write operation of DAC_SHSR1 ongoing: DAC_SHSR1 can be written

1:There is a write operation of DAC_SHSR1 ongoing: DAC_SHSR1 cannot be written

Bit 14 **CAL_FLAG1**: DAC channel1 calibration offset status

This bit is set and cleared by hardware

0: calibration trimming value is lower than the offset correction value

1: calibration trimming value is equal or greater than the offset correction value

Bit 13 **DMAUDR1**: DAC channel1 DMA underrun flag

This bit is set by hardware and cleared by software (by writing it to 1).

0: No DMA underrun error condition occurred for DAC channel1

1: DMA underrun error condition occurred for DAC channel1 (the currently selected trigger is driving DAC channel1 conversion at a frequency higher than the DMA service capability rate)

Bit 12 **DORSTAT1**: DAC channel1 output register status bit

This bit is set and cleared by hardware. It is applicable only when the DAC operates in double data mode.

0: DOR[11:0] is used actual DAC output

1: DORB[11:0] is used actual DAC output

Bit 11 **DAC1RDY**: DAC channel1 ready status bit

This bit is set and cleared by hardware.

0: DAC channel1 is not yet ready to accept the trigger nor output data

1: DAC channel1 is ready to accept the trigger or output data

Bits 10:0 Reserved, must be kept at reset value.

28.7.15 DAC calibration control register (DAC_CCR)

Address offset: 0x38

Reset value: 0x00XX 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4:0]				
											rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **OTRIM2[4:0]**: DAC channel2 offset trimming value

These bits are available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **OTRIM1[4:0]**: DAC channel1 offset trimming value

28.7.16 DAC mode control register (DAC_MCR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2[2:0]		
						rw	rw						rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HFSEL[1:0]		Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1[2:0]		
rw	rw					rw	rw						rw	rw	rw

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

Bit 25 **SINFORMAT2**: Enable signed format for DAC channel2

This bit is set and cleared by software.

0: Input data is in unsigned format

1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bit 24 **DMADOUBLE2**: DAC channel2 DMA double data mode

This bit is set and cleared by software.

0: DMA normal mode selected

1: DMA double data mode selected

Note: This bit is available only on dual-channel DACs. Refer to Section 28.3: DAC implementation.

Bits 23:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE2[2:0]:** DAC channel2 mode

These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN2 = 0 and bit CEN2 = 0 in the DAC_CR register). If EN2 = 1 or CEN2 = 1 the write operation is ignored.

They can be set and cleared by software to select the DAC channel2 mode:

– DAC channel2 in normal mode

000: DAC channel2 is connected to external pin with Buffer enabled

001: Reserved

010: DAC channel2 is connected to external pin with buffer disabled

011: Reserved

– DAC channel2 in sample and hold mode

100: DAC channel2 is connected to external pin with Buffer enabled

101: Reserved

110: DAC channel2 is connected to external pin with Buffer disabled

111: Reserved

Note: This register can be modified only when EN2 = 0.

Refer to Section 28.3: DAC implementation for the availability of DAC channel2.

Bits 15:14 **HFSEL[1:0]**: High frequency interface mode selection
00: High frequency interface mode disabled
01: High frequency interface mode enabled for AHB clock frequency > 80 MHz
10: High frequency interface mode enabled for AHB clock frequency >160 MHz
11: Reserved

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SINFORMAT1**: Enable signed format for DAC channel1
This bit is set and cleared by software.
0: Input data is in unsigned format
1: Input data is in signed format (2's complement). The MSB bit represents the sign.

Bit 8 **DMADOUBLE1**: DAC channel1 DMA double data mode
This bit is set and cleared by software.
0: DMA normal mode selected
1: DMA double data mode selected

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **MODE1[2:0]:** DAC channel1 mode
These bits can be written only when the DAC is disabled and not in the calibration mode (when bit EN1 = 0 and bit CEN1 = 0 in the DAC_CR register). If EN1 = 1 or CEN1 = 1 the write operation is ignored.
They can be set and cleared by software to select the DAC channel1 mode:
– DAC channel1 in normal mode
000: DAC channel1 is connected to external pin with Buffer enabled
001: Reserved
010: DAC channel1 is connected to external pin with Buffer disabled
011: Reserved
– DAC channel1 in sample & hold mode
100: DAC channel1 is connected to external pin with Buffer enabled
101: Reserved
110: DAC channel1 is connected to external pin with Buffer disabled
111: Reserved
Note: This register can be modified only when EN1 = 0.

28.7.17 DAC channel1 sample and hold sample time register (DAC_SHSR1)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE1[9:0]:** DAC channel1 sample time (only valid in sample and hold mode)

These bits can be written when the DAC channel1 is disabled or also during normal operation. in the latter case, the write can be done only when BWST1 of DAC_SR register is low, If BWST1 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

28.7.18 DAC channel2 sample and hold sample time register (DAC_SHSR2)

This register is available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 9:0 **TSAMPLE2[9:0]:** DAC channel2 sample time (only valid in sample and hold mode)
These bits can be written when the DAC channel2 is disabled or also during normal operation. in the latter case, the write can be done only when BWST2 of DAC_SR register is low, if BWST2 = 1, the write operation is ignored.

*Note: It represents the number of LSI/LSE clocks to perform a sample phase. Sampling time = (*TSAMPLE1[9:0] + 1) x *LSI/LSE clock period.*

28.7.19 DAC sample and hold time register (DAC_SHHR)

Address offset: 0x48

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 25:16 **THOLD2[9:0]:** DAC channel2 hold time (only valid in sample and hold mode).
Hold time = (THOLD[9:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **THOLD1[9:0]:** DAC channel1 hold time (only valid in sample and hold mode)
Hold time = (THOLD[9:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

28.7.20 DAC sample and hold refresh time register (DAC_SHRR)

Address offset: 0x4C

Reset value: 0x0001 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TREFRESH2[7:0]:** DAC channel2 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
These bits are available only on dual-channel DACs. Refer to *Section 28.3: DAC implementation*.
Note: This register can be modified only when EN2 = 0.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TREFRESH1[7:0]:** DAC channel1 refresh time (only valid in sample and hold mode)
Refresh time = (TREFRESH[7:0]) x LSI/LSE clock period
Note: This register can be modified only when EN1 = 0.

Note: These bits can be written only when the DAC channel is disabled and in normal operating mode (when bit ENx = 0 and bit CENx = 0 in the DAC_CR register). If ENx = 1 or CENx = 1 the write operation is ignored.

28.7.21 DAC register map

Table 288 summarizes the DAC registers.

Table 288. DAC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**DAC_CR**	Res.	CEN2	DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[2:0]		TSEL2[3:1]			TSEL2[0]	TEN2	EN2	Res.	CEN1	DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[3:1]			TSEL1[0]	TEN1	EN1
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**DAC_ SWTRGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWTRIG2	SWTRIG1
	Reset value																															0	0
0x08	**DAC_ DHR12R1**	Res.	Res.	Res.	Res.	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**DAC_ DHR12L1**	DACC1DHRB[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x10	**DAC_ DHR8R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC1DHRB[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**DAC_ DHR12R2**	Res.	Res.	Res.	Res.	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x18	**DAC_ DHR12L2**	DACC2DHRB[11:0]												Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				
0x1C	**DAC_ DHR8R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHRB[7:0]								DACC2DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**DAC_ DHR12RD**	Res.	Res.	Res.	Res.	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x24	**DAC_ DHR12LD**	DACC2DHR[11:0]												Res.	Res.	Res.	Res.	DACC1DHR[11:0]												Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0				

Table 288. DAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	**DAC_ DHR8RD**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DACC2DHR[7:0]								DACC1DHR[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**DAC_ DOR1**	Res.	Res.	Res.	Res.	DACC1DORB[11:0]												Res.	Res.	Res.	Res.	DACC1DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x30	**DAC_ DOR2**	Res.	Res.	Res.	Res.	DACC2DORB[11:0]												Res.	Res.	Res.	Res.	DACC2DOR[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x34	**DAC_SR**	BWST2	CAL_FLAG2	DMAUDR2	DORSTAT2	DAC2RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BWST1	CAL_FLAG1	DMAUDR1	DORSTAT1	DAC1RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0												0	0	0	0	0											
0x38	**DAC_CCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM2[4]	OTRIM2[3]	OTRIM2[2]	OTRIM2[1]	OTRIM2[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTRIM1[4]	OTRIM1[3]	OTRIM1[2]	OTRIM1[1]	OTRIM1[0]
	Reset value												X	X	X	X	X												X	X	X	X	X
0x3C	**DAC_MCR**	Res.	Res.	Res.	Res.	Res.	Res.	SINFORMAT2	DMADOUBLE2	Res.	Res.	Res.	Res.	Res.	MODE2 [2:0]			HFSEL[1]	HFSEL[0]	Res.	Res.	Res.	Res.	SINFORMAT1	DMADOUBLE1	Res.	Res.	Res.	Res.	Res.	MODE1 [2:0]		
	Reset value							0	0						0	0	0	0	0					0	0						0	0	0
0x40	**DAC_ SHSR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE1[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x44	**DAC_ SHSR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSAMPLE2[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x48	**DAC_ SHHR**	Res.	Res.	Res.	Res.	Res.	Res.	THOLD2[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	THOLD1[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	1							0	0	0	0	0	0	0	0	0	1
0x4C	**DAC_ SHRR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH2[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TREFRESH1[7:0]							
	Reset value									0	0	0	0	0	0	0	1									0	0	0	0	0	0	0	1
0x50-0x54	Reserved	Res.																															
0x58-0x60	Reserved	Res.																															
0x64-0x68	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

29 Voltage reference buffer (VREFBUF)

29.1 VREFBUF introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADCs, and also as voltage reference for external components through the VREF+ pin.

29.2 VREFBUF implementation

The table below describes the VREFBUF voltages typical values:

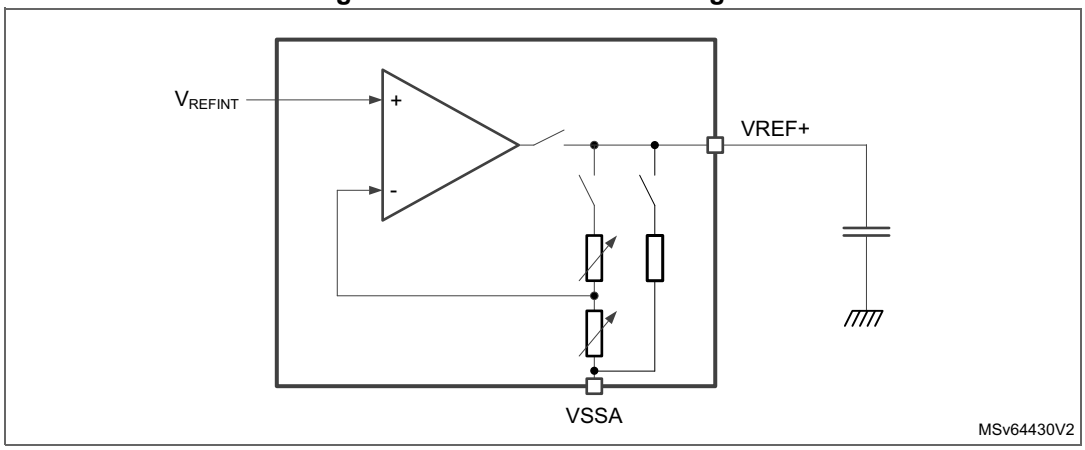
Table 289. VREFBUF typical values

Symbol	Value
VREFBUF0	2.5 V
VREFBUF1	2.048 V
VREFBUF2	1.8 V

Note: Refer to the product datasheet for more details.

29.3 VREFBUF functional description

Figure 278. VREFBUF block diagram



The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference V_{REFINT}. The VREFBUF supports three voltages, which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: VREFBUF0 voltage selected
- VRS = 001: VREFBUF1 voltage selected
- VRS = 010: VREFBUF2 voltage selected

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 290. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

29.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

29.5 VREFBUF registers

29.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: VREFBUF0 voltage selected.

VRS = 001: VREFBUF1 voltage selected.

VRS = 010: VREFBUF2 voltage selected.

Others: Reserved

Note: Refer to the product datasheet for each VREFBUFx voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.

0: V_{REF+} pin is internally connected to the voltage reference buffer output.

1: V_{REF+} pin is high impedance.

Refer to *Table 290: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.

0: Internal voltage reference mode disable (external voltage reference mode).

1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

29.5.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

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

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

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

Bits 5:0 **TRIM[5:0]**: Trimming code

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

Reset:

TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the flash memory during the production test.

VRS change:

TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the flash memory during the production test.

Write in TRIM[5:0]:

User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

29.5.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 291. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

30 Digital camera interface (DCMI)

30.1 DCMI introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

30.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw Bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

30.3 DCMI functional description

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (DCMI_D[13:0]) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

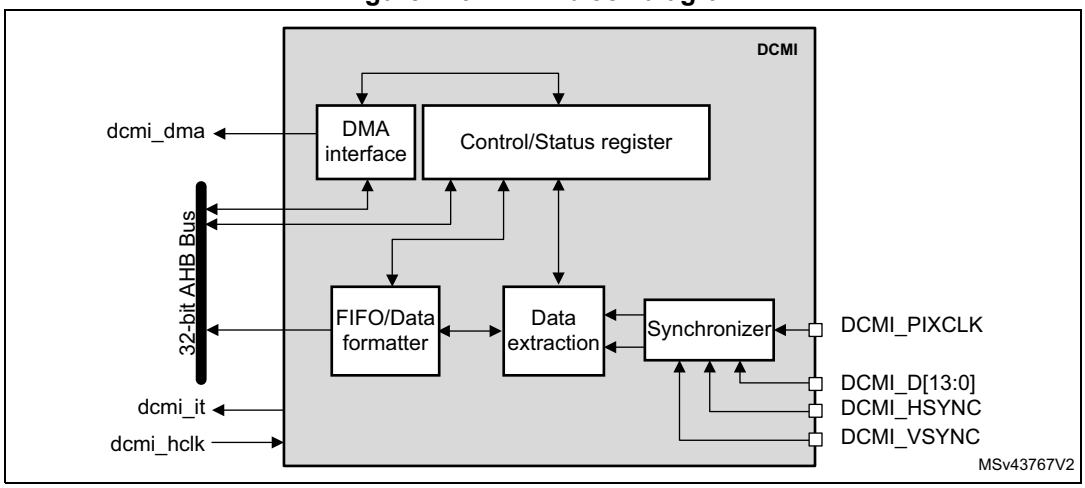
The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

30.3.1 DCMI block diagram

Figure 279 shows the DCMI block diagram.

Figure 279. DCMI block diagram



30.3.2 DCMI pins and internal signals

The following table shows DCMI pins.

Table 292. DCMI input/output pins

Mode	Pin name	Signal type	Description
8 bits 10 bits 12 bits 14 bits	DCMI_D[7:0] DCMI_D[9:0] DCMI_D[11:0] DCMI_D[13:0]	Inputs	DCMI data
DCMI_PIXCLK		Input	Pixel clock
DCMI_HSYNC		Input	Horizontal synchronization / Data valid
DCMI_VSYNC		Input	Vertical synchronization

The following table shows DCMI internal signals.

Table 293. DCMI internal input/output signals

Internal signal name	Signal type	Description
dcmi_dma	Output	DCMI DMA request
dcmi_it	Output	DCMI interrupt request
dcmi_hclk	Input	DCMI interface clock

30.3.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

30.3.4 DCMI DMA interface

The DMA interface is active when the CAPTURE bit of the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

30.3.5 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits of the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

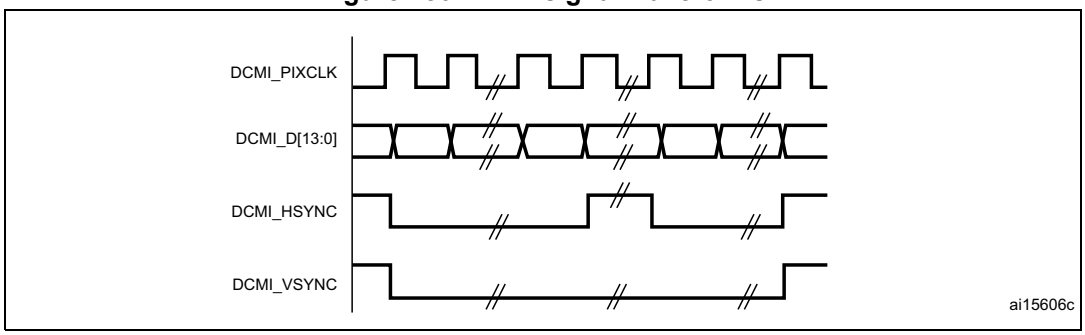
DCMI pins are shown in *Table 292*.

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame

Figure 280. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] = 00 in DCMI_CR the interface captures 8 LSBs at its input (DCMI_D[7:0]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. The table below gives an example of the positioning of captured data bytes in two 32-bit words.

Table 294. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10-bit data

When EDM[1:0] = 01 in DCMI_CR, the camera interface captures 10-bit data at its input DCMI_D[9:0] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits of the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 295. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 12-bit data at its input DCMI_D[11:0] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 296. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-bit data

When EDM[1:0] = 11 in DCMI_CR, the camera interface captures the 14-bit data at its input DCMI_D[13:0] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 297. Positioning of captured data bytes in 32-bit words (14-bit width)

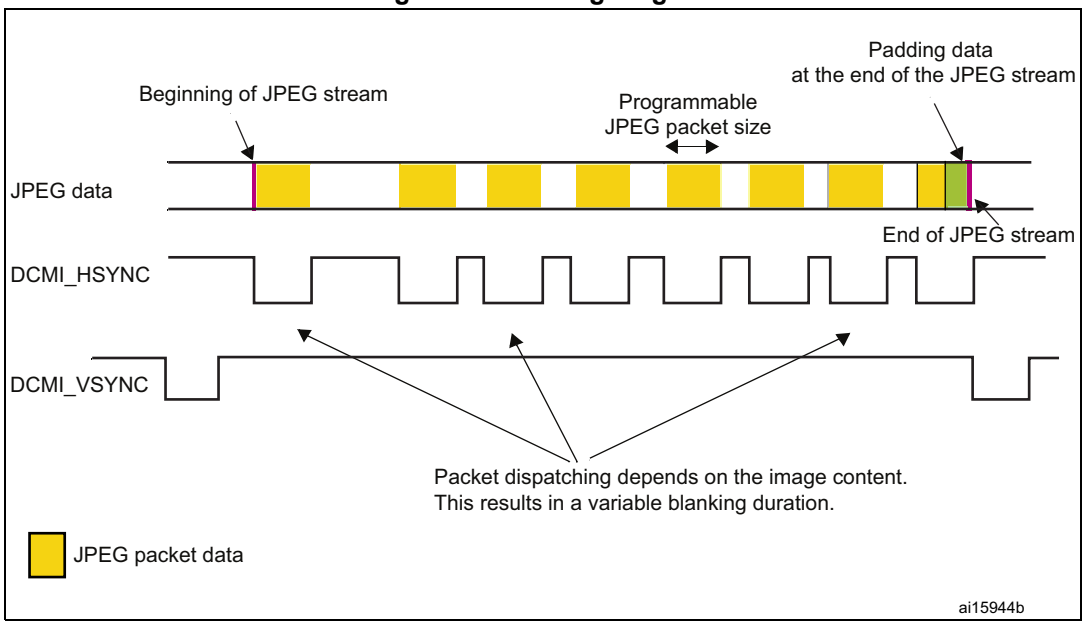
Byte address	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

30.3.6 DCMI synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits must be cleared).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. *Figure 281* shows the corresponding timing diagram.

Figure 281. Timing diagram



Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and

capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits must be cleared). For other data widths, this mode generates unpredictable results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

  Four embedded codes signal the following events
  - Frame start (FS)
  - Frame end (FE)
  - Line start (LS)
  - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see *Section 30.5.7: DCMI embedded synchronization code register (DCMI_ESCR)*).

  A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.
- Mode 1

  An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS $\leq$ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. A bit can therefore be selected to compare in the embedded code and

detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

30.3.7 DCMI capture modes

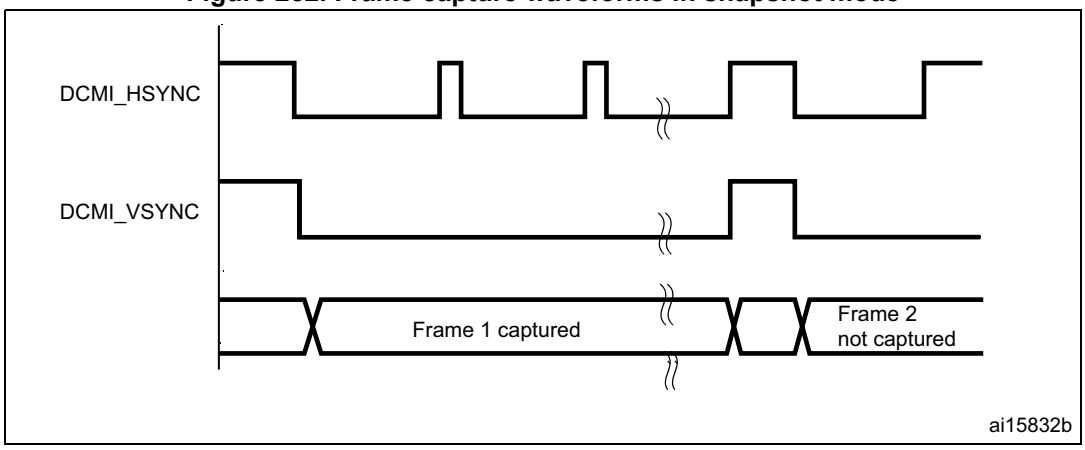
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = 1 of the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 282. Frame capture waveforms in snapshot mode

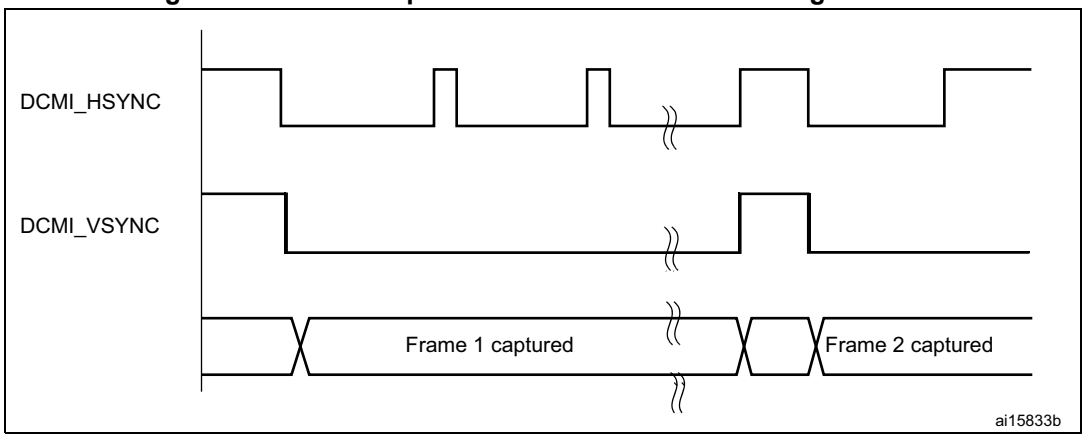


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = 0 in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 283. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

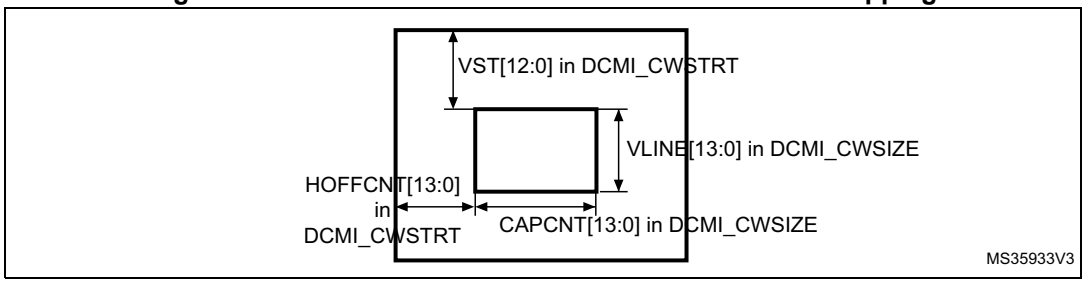
In continuous grab mode, the FCRC[1:0] bits in DCMI_CR can be configured to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

Note: In the hardware synchronization mode (ESS = 0 in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = 0 in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

30.3.8 DCMI crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

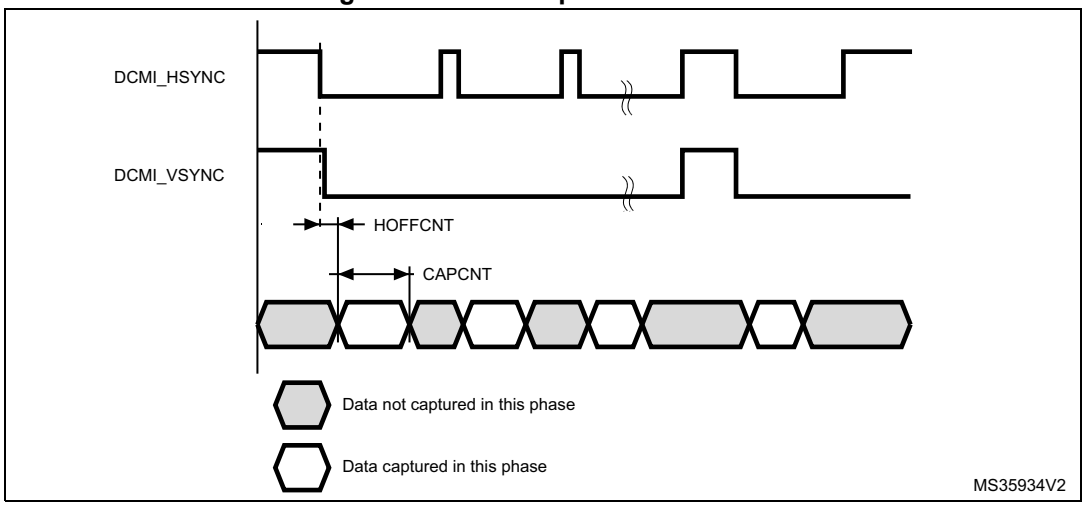
Figure 284. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 285. Data capture waveforms



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

30.3.9 DCMI JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit of the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4. This case must be carefully handled since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with zeros and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in JPEG format.

30.3.10 DCMI FIFO

A 8-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

30.3.11 DCMI data format description

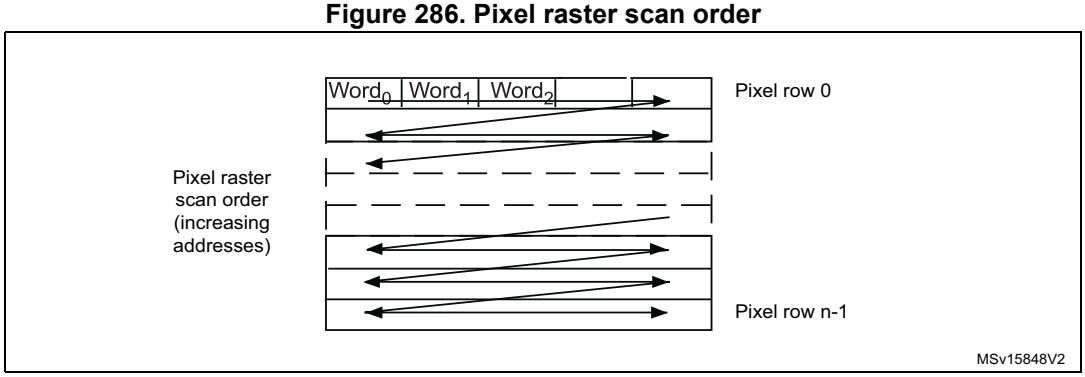
Data formats

The following types of data are supported:

- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred
- compressed data: JPEG

For monochrome, RGB and YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little-endian format is supported.

Figure 286. Pixel raster scan order



Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

The table below shows how the data are stored.

Table 298. Data storage in monochrome progressive video format

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G and B interleaved (such as BRGBRGBRG)
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats.

Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.

The 24 BPP (palletized format) and gray-scale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

The table below shows how the data are stored.

Table 299. Data storage in RGB progressive video format

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one buffer: Y, Cb and Cr interleaved (such as CbYCrYCbYCr)

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in the table below.

Table 300. Data storage in YCbCr progressive video format

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

YCbCr format - Y only

Characteristics:

- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only the luma component of each pixel, encoded in 8 bits, is stored as shown in *Table 301*.

The result is a monochrome image having the same resolution as the original YCbCr data.

Table 301. Data storage in YCbCr progressive video format - Y extraction mode

Byte address	31:24	23:16	15:8	7:0
0	Y n + 3	Y n + 2	Y n + 1	Y n
4	Y n + 7	Y n + 6	Y n + 5	Y n + 4

Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode is used to only store a half resolution image. It is selected through OELS and LSM control bits.

30.4 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. The table below gives the list of all interrupts.

Table 302. DCMI interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exits Sleep mode	Exists Stop and Standby modes
dcmi_it	End of line	LINE_RIS	LINE_IE	Set LINE_ISC	Yes	No
	End of frame capture	FRAME_RIS	FRAME_IE	Set FRAME_ISC	Yes	No
	Overrun of data reception	OVR_RIS	OVR_IE	Set OVR_ISC	Yes	No
	Synchronization frame	VSYNC_RIS	VSYNC_IE	Set VSYNC_ISC	Yes	No
	Detection of an error in the embedded synchronization frame detection	ERR_RIS	ERR_IE	Set ERR_ISC	Yes	No

30.5 DCMI registers

Refer to *Section 1.2 on page 104* for list of abbreviations used in register descriptions. All DCMI registers must be accessed as 32-bit words, otherwise a bus error occurs.

30.5.1 DCMI control register (DCMI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OELS:** Odd/Even Line Select (Line Select Start)

This bit works in conjunction with the LSM field (LSM = 1).

0: Interface captures first line after the frame start, second one being dropped.

1: Interface captures second line from the frame start, first one being dropped.

Bit 19 **LSM:** Line Select mode

0: Interface captures all received lines.

1: Interface captures one line out of two.

Bit 18 **OEBS:** Odd/Even Byte Select (Byte Select Start)

This bit works in conjunction with BSM field (BSM $\neq$ 00).

0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped.

1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped.

Bits 17:16 **BSM[1:0]:** Byte Select mode

00: Interface captures all received data.

01: Interface captures every other byte from the received data.

10: Interface captures one byte out of four.

11: Interface captures two bytes out of four.

Note: This mode only works for EDM[1:0] = 00. For all other EDM values, this field must be programmed to the reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** DCMI enable

0: DCMI disabled

1: DCMI enabled

Note: The DCMI configuration registers must be programmed correctly before enabling this bit.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode

00: Interface captures 8-bit data on every pixel clock.

01: Interface captures 10-bit data on every pixel clock.

10: Interface captures 12-bit data on every pixel clock.

11: Interface captures 14-bit data on every pixel clock.

Bits 9:8 **FCRC[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.

00: All frames are captured.

01: Every alternate frame captured (50% bandwidth reduction)

10: One frame out of four captured (75% bandwidth reduction)

11: reserved

Bit 7 **VSPOL:** Vertical synchronization polarity

This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.

0: DCMI_VSYNC active low

1: DCMI_VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity

This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.

0: DCMI_HSYNC active low

1: DCMI_HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity

This bit configures the capture edge of the pixel clock.

0: Falling edge active

1: Rising edge active

Bit 4 **ESS:** Embedded synchronization select

0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.

1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.

Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.

This bit is disabled in JPEG mode.

Bit 3 **JPEG:** JPEG format

0: Uncompressed video format

1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2 **CROP:** Crop feature

0: The full image is captured. In this case the total number of bytes in an image frame must be a multiple of four.

1: Only the data inside the window specified by the crop register is captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE:** Capture enable

0: Capture disabled

1: Capture enabled

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the first frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit is effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers must be programmed correctly before enabling this bit.

30.5.2 DCMI status register (DCMI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

Bit 2 **FNE**: FIFO not empty

This bit gives the status of the FIFO.
1: FIFO contains valid data.
0: FIFO empty

Bit 1 **VSYNC**: Vertical synchronization

This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active frame
1: synchronization between frames
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**: Horizontal synchronization

This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active line
1: synchronization between lines
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

30.5.3 DCMI raw interrupt status register (DCMI_RIS)

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
											r	r	r	r	r

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

Bit 4 **LINE_RIS**: Line raw interrupt status

This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by setting the LINE_ISC bit of the DCMI_ICR register.

Bit 3 **VSYNC_RIS**: DCMI_VSYNC raw interrupt status

This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by setting the VSYNC_ISC bit of the DCMI_ICR register.

Bit 2 **ERR_RIS**: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by setting the ERR_ISC bit of the DCMI_ICR register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_RIS**: Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

The bit is cleared by setting the OVR_ISC bit of the DCMI_ICR register.

Bit 0 **FRAME_RIS**: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

The bit is cleared by setting the FRAME_ISC bit of the DCMI_ICR register.

30.5.4 DCMI interrupt enable register (DCMI_IER)

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 4 **LINE_IE**: Line interrupt enable
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received.

Bit 3 **VSYNC_IE**: DCMI_VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE**: Synchronization error interrupt enable
0: No interrupt generation
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE**: Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE**: Capture complete interrupt enable
0: No interrupt generation
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

30.5.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bit 4 **LINE_MIS**: Line masked interrupt status
This bit gives the status of the masked line interrupt.
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS**: VSYNC masked interrupt status

This bit gives the status of the masked VSYNC interrupt.

0: No interrupt is generated on DCMI_VSYNC transitions.

1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.

The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS**: Synchronization error masked interrupt status

This bit gives the status of the masked synchronization error interrupt.

0: No interrupt is generated on a synchronization error.

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS**: Overrun masked interrupt status

This bit gives the status of the masked overflow interrupt.

0: No interrupt is generated on overrun.

1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS**: Capture complete masked interrupt status

This bit gives the status of the masked capture complete interrupt

0: No interrupt is generated after a complete capture.

1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

30.5.6 DCMI interrupt clear register (DCMI_ICR)

The DCMI_ICR register is write-only. Setting a bit of this register clears the corresponding flag in the DCMI_RIS and DCMI_MIS registers. Writing 0 has no effect.

Address offset: 0x14

Reset value: 0x0000 0000

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

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

Bit 4 **LINE_ISC**: line interrupt status clear

Setting this bit clears the LINE_RIS flag in the DCMI_RIS register.

Bit 3 **VSYNC_ISC**: Vertical Synchronization interrupt status clear

Setting this bit clears the VSYNC_RIS flag in the DCMI_RIS register.

Bit 2 **ERR_ISC**: Synchronization error interrupt status clear

Setting this bit clears the ERR_RIS flag in the DCMI_RIS register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC**: Overrun interrupt status clear
Setting this bit clears the OVR_RIS flag in the DCMI_RIS register.

Bit 0 **FRAME_ISC**: Capture complete interrupt status clear
Setting this bit clears the FRAME_RIS flag in the DCMI_RIS register.

30.5.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bits 31:24 **FEC[7:0]**: Frame end delimiter code
This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.
If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code
This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code
This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code
This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.
If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the first occurrence of LSC after an FEC code is interpreted as a start of frame delimiter.

30.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.
0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.
1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data.

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.
0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data.
1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data.

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.
0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data.
1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data.

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data.

30.5.9 DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **VST[12:0]**: Vertical start line count

The image capture starts with this line number. Previous line data are ignored.
0x0000: line 1
0x0001: line 2
0x0002: line 3
....

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

Bits 13:0 **HOFFCNT[13:0]**: Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

30.5.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **VLINE[13:0]**: Vertical line count

This value gives the number of lines to be captured from the starting point.
0x0000: 1 line
0x0001: 2 lines
0x0002: 3 lines
....

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

Bits 13:0 **CAPCNT[13:0]**: Capture count

This value gives the number of pixel clocks to be captured from the starting point on the same line. It value must corresponds to word-aligned data for different widths of parallel interfaces.
0x0000: 1 pixel
0x0001: 2 pixels
0x0002: 3 pixels
....

30.5.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 8-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE1[7:0]								BYTE0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

30.5.12 DCMI register map

Table 303. DCMI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]		Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value												0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC[7:0]								LEC[7:0]								LSC[7:0]								FSC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DCMI_ESUR	FEU[7:0]								LEU[7:0]								LSU[7:0]								FSU[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DCMI_CWSTRT	Res.	Res.	Res.	VST[12:0]													Res.	Res.	HOFFCNT[13:0]													
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DCMI_CWSIZE	Res.	Res.	VLINE[13:0]														Res.	Res.	CAPCNT[13:0]													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DCMI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

31 Parallel synchronous slave interface (PSSI)

The PSSI peripheral and the DCMI (digital camera interface) use the same circuitry. As a result, these two peripherals cannot be used at the same time: when using the PSSI, the DCMI registers cannot be accessed, and vice-versa.

In addition, the PSSI and the DCMI share the same alternate functions and interrupt vector (see *Section 31.3.2: PSSI pins and internal signals*).

31.1 PSSI introduction

The PSSI is a generic synchronous 8/16-bit parallel data input/output slave interface. It enables the transmitter to send a data valid signal that indicates when the data is valid, and the receiver to output a flow control signal that indicates when it is ready to sample the data.

31.2 PSSI main features

The PSSI peripheral main features are the following:

- Slave mode operation
- 8-bit or 16-bit parallel data input or output
- 8-word (32-byte) FIFO
- Data enable (PSSI_DE) and Ready (PSSI_RDY) alternate function

  When selected, these signals can either enable the transmitter to indicate when the data is valid, allow the receiver to indicate when it is ready to sample the data, or both.

31.3 PSSI functional description

The PSSI is a synchronous parallel slave interface that can send or receive high-speed data flows. It consists of up to 16 data lines (PSSI_D[15:0]) plus a clock line (PSSI_PDCK). The clock polarity can be configured so that data can be captured or transmitted on either the clock rising or falling edge.

Usually, a general-purpose DMA channel is used to pass 32-bit packed data via the data register (PSSI_DR).

The data flow can either be continuous or synchronized by hardware using the optional PSSI_DE (Data enable), and PSSI_RDY (Ready) signals.

Figure 287 shows the PSSI block diagram.

31.3.1 PSSI block diagram

Figure 287. PSSI block diagram

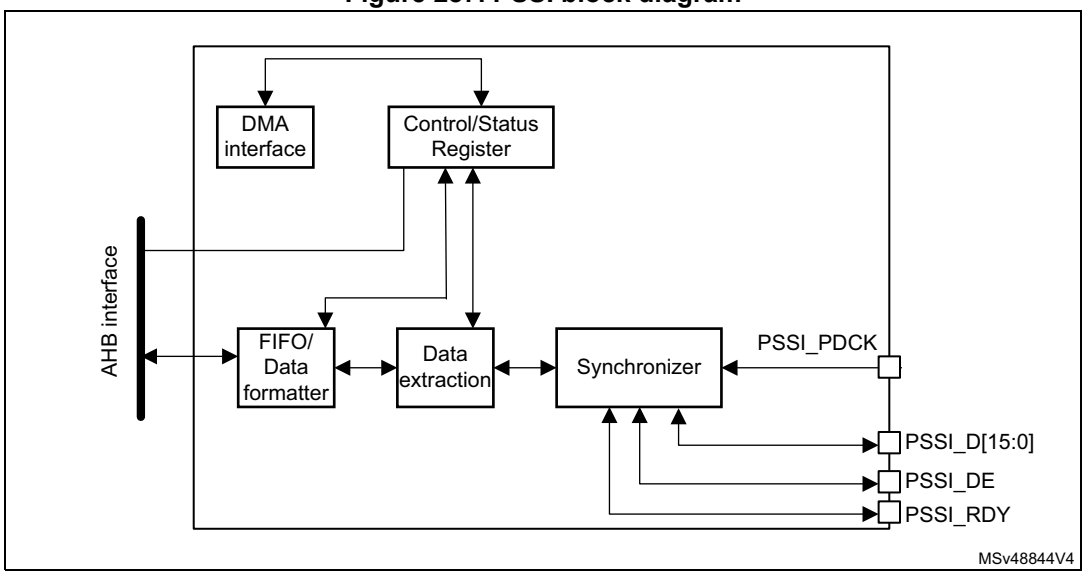
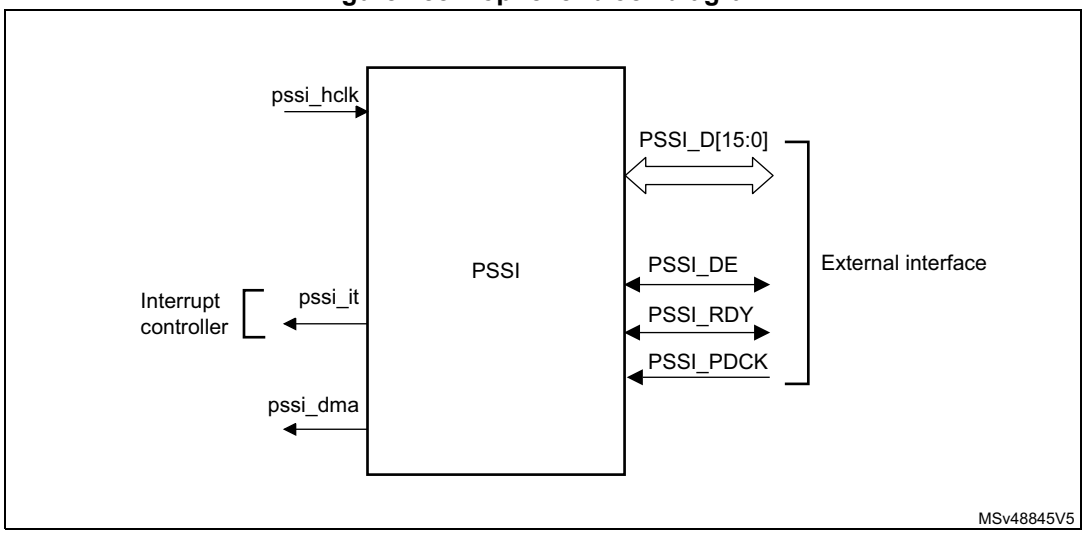


Figure 288. Top-level block diagram



31.3.2 PSSI pins and internal signals

The PSSI interface is composed of 19 pins, though nine signals are enough to transfer parallel data. *Table 304* shows the PSSI pins.

When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, the alternate functions and the interrupt vector are associated with the PSSI. Otherwise, they are associated with the DCMI. The DCMI ENABLE bit (bit 15 of DCMI_CR) and the PSSI ENABLE bit (bit 14 of PSSI_CR) must not be set to 1 at the same time. As an example, if a GPIO is configured to use the alternate function PSSI_PDCK/DCMI_PIXCK, it is the PSSI_PDCK function which becomes active if PSSI_CR/ENABLE is set to 1.

Table 304. PSSI input/output pins

PSSI signal name	DCMI signal it is shared with	Signal type	Description
PSSI_PDCK	DCMI_PIXCK	Input	Parallel data clock input
PSSI_D[15:0]	DCMI_D[13:0]	Input/output	Data output when transmitting, data input when receiving
PSSI_DE	DCMI_HSYNC	Input/output	Data enable signal
PSSI_RDY	DCMI_VSYNC	Input/output	Ready signal

Table 305 shows the PSSI internal input/output signals.

Table 305. PSSI internal input/output signals

Internal signal name	Signal type	Description
pssi_it	Output	Interrupt
pssi_dma	Output	DMA request
pssi_hclk	Input	AHB clock

31.3.3 PSSI clock

The AHB clock frequency must be at least 2.5 times higher than the PSSI_PDCK frequency. At frequency ratios lower than 2.5, data might be corrupted or lost during transfers.

Data transfers are synchronous with PSSI_PDCK. The PSSI_PDCK polarity can be configured as follows, through CKPOL bit (bit 5 of PSSI_CR):

- When CKPOL = 0
  - Input pins are sampled on PSSI_PDCK falling edge
  - Output pins are driven on PSSI_PDCK rising edge
- When CKPOL = 1
  - Input pins are sampled on PSSI_PDCK rising edge
  - Output pins are driven on PSSI_PDCK falling edge

31.3.4 PSSI data management

Data direction

The direction of data transfers is configured through the OUTEN control bit (bit 31 of PSSI_CR):

- When OUTEN is cleared to 0 (default setting), the PSSI operates in receive mode and the data is input on the data pins.
- When OUTEN is set to 1, the peripheral operates in transmit mode and the data is output on the data pins.

OUTEN can be modified only when the ENABLE bit is cleared to 0.

Data register and DMA

Data are transferred from/to the FIFO using the PSSI_DR data register:

- In receive mode, data must be read from the FIFO by reading PSSI_DR.
- In transmit mode, data must be written to the FIFO by writing into PSSI_DR.

Word (32-bit) accesses to PSSI_DR and half-word (16-bit) accesses to PSSI_DR[15:0] are permitted in all modes. Byte (8-bit) accesses to PSSI_DR[7:0] are permitted only when the PSSI is configured to transfer 8 bits at a time (EDM=00 in the PSSI_CR register).

To reduce the load on the CPU, it is recommended to use the DMA to transfer data from/to the PSSI FIFO. When it is used, the DMA must be configured to transfer data via the PSSI_DR register. Using 32-bit transfers optimizes bandwidth and reduces the bus load. However, 8-bit and 16-bit transfers are also permitted.

To use the DMA, set the PSSI DMA enable bit (DMAEN in PSSI_CR) to 1 (default setting). When DMAEN is set to 1, a DMA transfer is initiated when the FIFO is ready for a 32-bit transfer (four valid bytes in receive mode or four empty bytes in transmit mode). As a result, in receive mode, no DMA transfers are initiated if there are three bytes or fewer in the FIFO, even if the DMA is configured to perform 8-bit transfers.

The RTT4B and RTT1B status bits (PSSI_SR) are useful when the CPU directly perform transfers to and from the FIFO. RTT4B set to 1 indicates that the FIFO is ready to transfer four bytes: at least four valid bytes in the FIFO in receive mode or at least four free bytes in transmit mode. RTT1B set to 1 indicates that the FIFO is ready to transfer one byte: at least one valid byte in the FIFO in receive mode or at least one free byte in transmit mode.

8-bit data

The PSSI parallel interface can transfer either 8-bit (using D[7:0]) or 16-bit data (using D[15:0]) depending on the EDM[1:0] control bits (bits 11:10 of PSSI_CR). If the 8-bit configuration is selected (EDM[1:0] set to 00), the unused D[15:0] pins can be used for GPIO or other functions.

When EDM[1:0] in PSSI_CR are programmed to 00, the interface transfers 8 bits using the D[7:0] pins. In this case, D[15:8] are not used and four PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant byte (bits 7:0) correspond to the first byte transferred, and the most-significant byte (bits 31:28) corresponds to the forth byte transferred. *Table 306* illustrates the positioning of the data bytes in two 32-bit words.

Table 306. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

16-bit data

When EDM[1:0] in PSSI_CR are programmed to 11, the interface transfers 16 bits using the D[15:0] pins. In this case, two PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant half word (bits 15:0) correspond to the first half word transferred, and the most-significant half-word (bits 31:16) corresponds to the second half word transferred. *Table 307* illustrates the positioning of the data in two 32-bit words.

Table 307. Positioning of captured data bytes in 32-bit words (16-bit width)

Byte address	31:16	15:0
0	D_{n+1}[15:0]	D_n[15:0]
4	D_{n+3}[15:0]	D_{n+2}[15:0]

FIFO data buffer and error conditions

An eight-word FIFO helps improving performance and avoids overruns and underruns.

If the ready signal (PSSI_RDY) is disabled in receive mode, an overrun error is generated when a clock active edge occurs when the FIFO is full. In this case, the input data is lost.

If the data enable signal (PSSI_DE) is disabled in transmit mode, an underrun error is generated when a clock active edge occurs when the FIFO is empty. In this case, unpredictable data are output.

The OVR_RIS status bit indicates that either an overrun or an underrun occurred. An interrupt can be generated when these events occur.

31.3.5 PSSI optional control signals

Data Enable (PSSI_DE) alternate function input

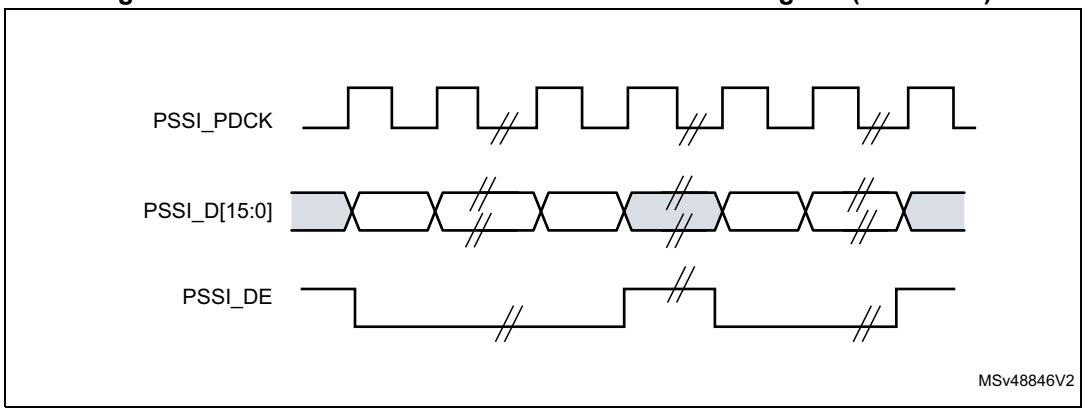
The data enable signal, PSSI_DE, is an optional signal. It is driven by the data source/transmitter in order to indicate that the data is valid to be transferred during the current cycle. When PSSI_DE is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG (bits 20:18 of PSSI_CR) control bits. PSSI_DE polarity is configured through DEPOL control bit (bit 6 of PSSI_CR). PSSI_DE is active low when DEPOL is cleared to 0, and high when DEPOL is et to 1.

The direction of the PSSI_DE signal is defined by the OUTEN value. It is the same as the data direction.

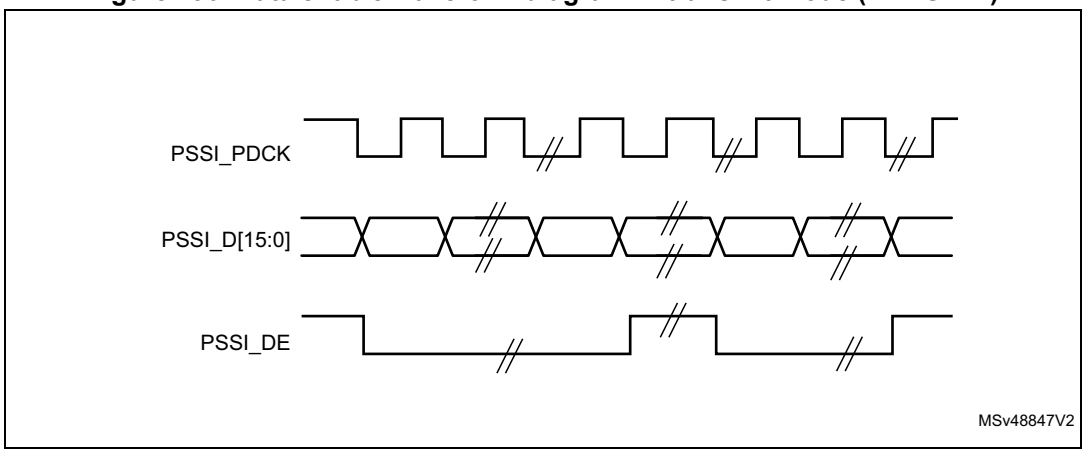
If the PSSI_DE alternate function input is enabled (through DERDYCFG) in receive mode (OUTEN cleared to 0), the PSSI samples PSSI_DE on the same PSSI_PDCK edge as the one used for sampling the data (D[15:0]). If PSSI_DE is active, the sampled data is saved in the FIFO. Otherwise, the sampled data is considered invalid and discarded. The transmitting device can use PSSI_DE as a data valid signal, driving it inactive when the data in the current cycle is not valid. This flow control function allows avoiding underrun errors.

Figure 289. Data enable in receive mode waveform diagram (CKPOL=0)



If the PSSI_DE alternate output function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI drives PSSI_DE on the same PSSI_PDCK edge that the one used to drive the data (D[15:0]). If a new 8 or 16-bit data (as programmed in the EDM[1:0] control bits in PSSI_CR) is available for transmission in the internal FIFO, this data is output on the data outputs (D[15:0]) and the PSSI_DE output becomes active on the current PSSI_PDCK edge. Otherwise (if the TX FIFO is empty), the D[15:0] outputs remains unchanged on the next clock edge and the PSSI_DE output becomes inactive.

Figure 290. Data enable waveform diagram in transmit mode (CKPOL=1)



Ready (PSSI_RDY) alternate function output

The ready signal, PSSI_RDY, is an optional signal. It is driven by the receiving device and indicates whether data is being accepted in the current cycle. When PSSI_RDY is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

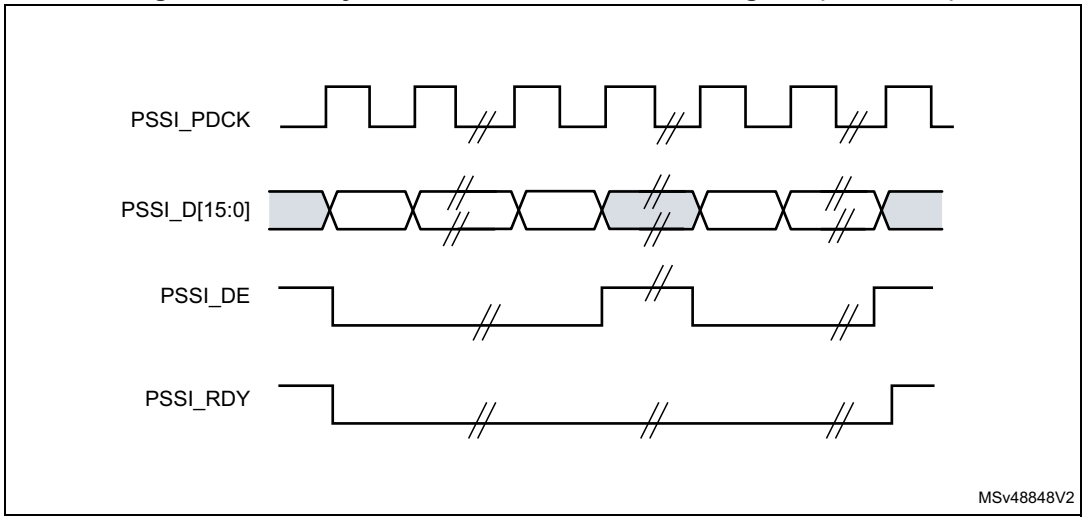
This alternate function signal can be enabled using the DERDYCFG control bits (bits 20:18 of PSSI_CR). PSSI_RDY polarity is configured through the RDYPOL control bit (bit 6 of PSSI_CR). PSSI_RDY is active low when RDYPOL is cleared to 0, and high when RDYPOL set to 1.

The direction of the PSSI_RDY signal is defined by the OUTEN (bit 31 of PSSI_CR). It is set in the opposite direction compared to the PSSI_DE and data signals.

If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples

the data (D[15:0]). If the FIFO has enough free space to receive more data, the PSSI drives the PSSI_RDY signal active. Otherwise, if the FIFO is full and cannot accept more data, the PSSI drives the PSSI_RDY signal inactive. The transmitting device must repeat the current data in the next cycle when it detects that PSSI_RDY is inactive. This flow control function allows the PSSI to avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Figure 291. Ready in receive mode waveform diagram (CKPOL=0)



If the PSSI_RDY alternate input function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI samples the PSSI_RDY signal on the opposite PSSI_PDCK edge to the one at which D[15:0] are driven. If the PSSI_RDY signal is inactive, the PSSI keeps the same data (D[15:0]) and PSSI_DE signals that valid data are available during the next PSSI_PDCK clock cycle. Otherwise, if PSSI_RDY signal is sampled as active, the next data from the TX FIFO (if available) is output on the data outputs (D[15:0]). If no new data are available in the TX FIFO, the PSSI keeps the data output values and outputs the PSSI_DE signal as inactive (if enabled).

The receiving device uses the PSSI_RDY to control the data flow and avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Bidirectional PSSI_DE/PSSI_RDY signal

A single pin can be used for both data enable (PSSI_DE) and ready (PSSI_RDY) functions if DEPOL and RDYPOL are both set to 1 and DERDYCFG is set to 111 or 100 in the PSSI_CR register. In this case, the GPIO corresponding to selected alternate function (PSSI_DE when DERDYCFG=111 or PSSI_RDY when DERDYCFG=100) must be configured as open-drain. The other device must also be configured to drive the line as open-drain, and a weak pull-up must be applied to the line.

The signal thus becomes bidirectional. If either the sender drives the line low (to indicate that the data is not valid) or the receiver drives the line low (to indicate that it is not sampling the current data), then both devices know that the data is not being transferred in the current cycle.

Figure 292. Bidirectional PSSI_DE/PSSI_RDY waveform

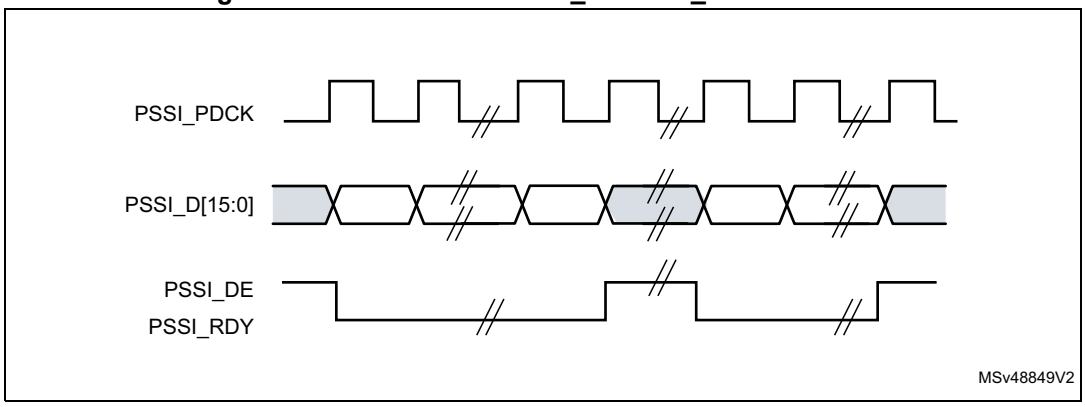
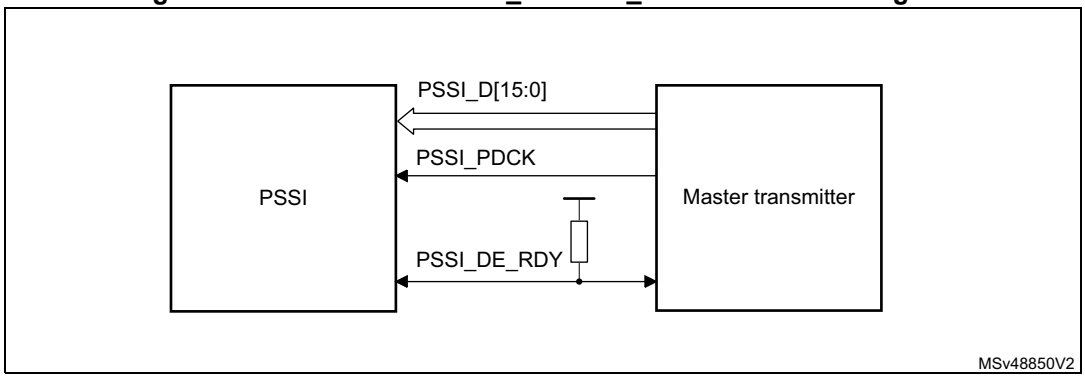


Figure 293. Bidirectional PSSI_DE/PSSI_RDY connection diagram



31.4 PSSI interrupts

The PSSI generates only one interrupt (IT_OVR). It is consequently equivalent to the global interrupt (pssi_it). Refer to *Table 308* for the list of interrupts.

The PSSI and the DCMI share the same interrupt vector. When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, these interrupts are triggered by the PSSI. Otherwise, they are controlled by the DCMI.

The DCMI ENABLE bit (bit 14 of DCMI_CR) and PSSI ENABLE bit must not be set to 1 at the same time.

Table 308. PSSI interrupt requests

Interrupt acronym	Shared with DCMI	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from low-power mode
IT_OVR	IT_OVR	indicates overrun in receive mode or underrun in transmit mode	OVR_RIS	OVR_IE	OVR_ISC	NA

31.5 PSSI registers

An 8-bit write or a 16-bit write operation to any PSSI register besides PSSI_DR, results in a bus error. 32-bit read and write operations are permitted.

31.5.1 PSSI control register (PSSI_CR)

Address offset: 0x00

Reset value: 0x4000 0000

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

Bit 31 **OUTEN:** Data direction selection bit

0: Receive mode: data is input synchronously with PSSI_PDCK

1: Transmit mode: data is output synchronously with PSSI_PDCK

Bit 30 **DMAEN:** DMA enable bit

0: DMA transfers are disabled. The user application can directly access the PSSI_DR register when DMA transfers are disabled.

1: DMA transfers are enabled (default configuration). A DMA channel in the general-purpose DMA controller must be configured to perform transfers from/to PSSI_DR.

Bits 29:21 Reserved, must be kept at reset value.

Bits 20:18 **DERDYCFG[2:0]:** Data enable and ready configuration

000: PSSI_DE and PSSI_RDY both disabled

001: Only PSSI_RDY enabled

010: Only PSSI_DE enabled

011: Both PSSI_RDY and PSSI_DE alternate functions enabled

100: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_RDY pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 1239*)

101: Only PSSI_RDY function enabled, but mapped to PSSI_DE pin

110: Only PSSI_DE function enabled, but mapped to PSSI_RDY pin

111: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_DE pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 1239*)

When the PSSI_RDY function is mapped to the PSSI_DE pin (settings 101 or 111), it is still the RDYPOL bit which determines its polarity. Similarly, when the PSSI_DE function is mapped to the PSSI_RDY pin (settings 110 or 111), it is still the DEPOL bit which determines its polarity.

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** PSSI enable
0: PSSI disabled
1: PSSI enabled
The contents of the FIFO are flushed when ENABLE is cleared to 0.

Note: When ENABLE=1, the content of PSSI_CR must not be changed, except for the ENABLE bit itself. All configuration bits can change as soon as ENABLE changes from 0 to 1.

The DMA controller and all PSSI configuration registers must be programmed correctly before setting the ENABLE bit to 1.

The ENABLE bit and the DCMI ENABLE bit (bit 15 of DCMI_CR) must not be set to 1 at the same time.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode
00: Interface captures 8-bit data on every parallel data clock
01: Reserved, must not be selected
10: Reserved, must not be selected
11: The interface captures 16-bit data on every parallel data clock

Bit 9 Reserved, must be kept at reset value.

Bit 8 **RDYPOL:** Ready (PSSI_RDY) polarity
This bit indicates the level on the PSSI_RDY pin when the data are not valid on the parallel interface.
0: PSSI_RDY active low (0 indicates that the receiver is ready to receive)
1: PSSI_RDY active high (1 indicates that the receiver is ready to receive)

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DEPOL:** Data enable (PSSI_DE) polarity
This bit indicates the level on the PSSI_DE pin when the data are not valid on the parallel interface.
0: PSSI_DE active low (0 indicates that data is valid)
1: PSSI_DE active high (1 indicates that data is valid)

Bit 5 **CKPOL:** Parallel data clock polarity
This bit configures the capture edge of the parallel clock or the edge used for driving outputs, depending on OUTEN.
0: Falling edge active for inputs or rising edge active for outputs
1: Rising edge active for inputs or falling edge active for outputs.

Bits 4:0 Reserved, must be kept at reset value.

31.5.2 PSSI status register (PSSI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTT1B	RTT4B	Res.	Res.
												r	r		

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

Bit 3 **RTT1B:** FIFO is ready to transfer one byte

1: FIFO is ready for a one byte transfer. In receive mode, this means that at least one valid data byte is in the FIFO. In transmit mode, this means that there is at least one byte free in the FIFO.

0: FIFO is not ready for a 1-byte transfer

Bit 2 **RTT4B:** FIFO is ready to transfer four bytes

1: FIFO is ready for a four-byte (32-bit) transfer. In receive mode, this means that at least four valid data bytes are in the FIFO. In transmit mode, this means that there are at least four bytes free in the FIFO.

0: FIFO is not ready for a four-byte transfer

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

31.5.3 PSSI raw interrupt status register (PSSI_RIS)

Address offset: 0x08

Reset value: 0x0000 0000

PSSI_RIS gives the raw interrupt status. This register is read-only. When read, it returns the status of the corresponding interrupt before masking with the PSSI_IER register value.

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

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

Bit 1 **OVR_RIS:** Data buffer overrun/underrun raw interrupt status

0: No overrun/underrun occurred

1: An overrun/underrun occurred: overrun in receive mode, underrun in transmit mode. This bit is cleared by writing a 1 to the OVR_ISC bit in PSSI_ICR.

Bit 0 Reserved, must be kept at reset value.

31.5.4 PSSI interrupt enable register (PSSI_IER)

Address offset: 0x0C

Reset value: 0x0000 0000

The PSSI_IER register is used to enable interrupts. When one of the PSSI_IER bits is set, the corresponding interrupt is enabled. This register is accessible both in read and write modes.

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

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

Bit 1 **OVR_IE:** Data buffer overrun/underrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if either an overrun or an underrun error occurred.

Bit 0 Reserved, must be kept at reset value.

31.5.5 PSSI masked interrupt status register (PSSI_MIS)

This PSSI_MIS register is read-only. When read, it returns the current masked status value of the corresponding interrupt (depending on the value in PSSI_IER). A bit in this register is set if the corresponding enable bit in PSSI_IER is set and the corresponding bit in PSSI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bit 1 **OVR_MIS:** Data buffer overrun/underrun masked interrupt status
This bit is set to 1 only when PSSI_IER/OVR_IE and PSSI_RIS/OVR_RIS are both set to 1.
0: No interrupt is generated when an overrun/underrun error occurs
1: An interrupt is generated if there is either an overrun or an underrun error and the OVR_IE bit is set in PSSI_IER.

Bit 0 Reserved, must be kept at reset value.

31.5.6 PSSI interrupt clear register (PSSI_ICR)

Address offset: 0x14

Reset value: 0x0000 0000

The PSSI_ICR register is write-only. Writing a 1 into a bit of this register clears the corresponding bit in the PSSI_RIS and PSSI_MIS registers. Writing a 0 has no effect. Reading this register always gives zeros.

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

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

Bit 1 **OVR_ISC:** Data buffer overrun/underrun interrupt status clear
Writing this bit to 1 clears the OVR_RIS bit in PSSI_RIS.

Bit 0 Reserved, must be kept at reset value.

31.5.7 PSSI data register (PSSI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

In receive mode (OUTEN = 0), the DMA controller must read the received data from this register. Write operations to PSSI_DR result in an error response. When more bytes than the number of valid bytes are read in the FIFO, the invalid bytes return zeros.

In transmit mode (OUTEN = 1), the DMA controller must write the data to be transmitted into this register. Read operations to PSSI_DR result in an error response.

32-bit, 16-bit, and 8-bit accesses are all supported for PSSI_DR. For instance, 16-bit read/write operations remove/add two bytes from/to the FIFO. However, 8-bit accesses are permitted only when the PSSI is configured to transfer 8 data bits at a time (EDM=00 in PSSI_CR). 8-bit accesses to PSSI_DR when EDM is not set to 0 result in an error response.

All accesses must include byte 0: 8-bit accesses must be performed to bits 7 to 0 and 16-bit accesses from bits 15 to 0. Accesses that do not include byte 0 results in an error response.

Accessing PSSI_DR when ENABLE bit in PSSI_CR is set to 0 results in an error response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE1[7:0]								BYTE0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

31.5.8 PSSI register map

Table 309. PSSI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PSSI_CR	OUTEN	DMAEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERDYCFG			Res.	Res.	Res.	ENABLE[2:0]			EDM		Res.	RDYPOL	Res.	DEPOL	CKPOL	Res.	Res.	Res.	Res.	Res.
	Reset value	0	1										0	0	0				0			0	0		0		0	0					
0x04	PSSI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTT1B	RTT4B	Res.	Res.
	Reset value																													0	0		
0x08	PSSI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_RIS	Res.
	Reset value																															0	
0x0C	PSSI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.am	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_IE	Res.
	Reset value																															0	
0x10	PSSI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_MIS	Res.
	Reset value																															0	
0x14	PSSI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_ISC	Res.
	Reset value																															0	
0x18-0x24	Reserved	Res.																															
0x28	PSSI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

32 True random number generator (RNG)

32.1 RNG introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

32.2 RNG main features

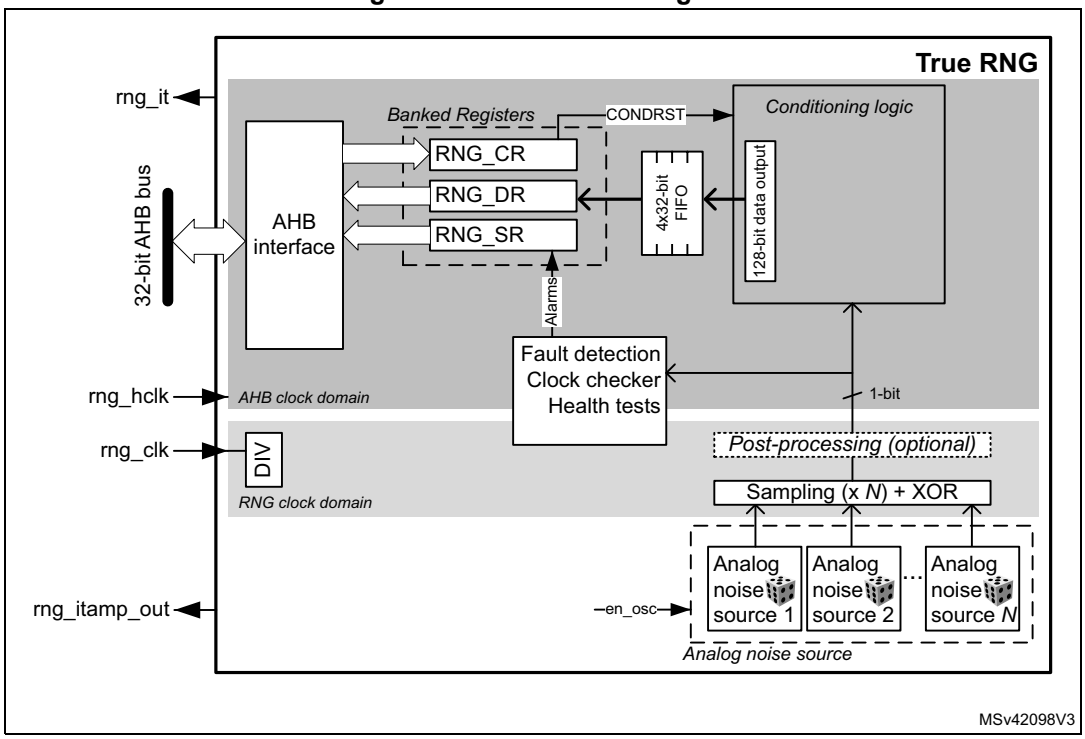
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a nondeterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- It embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

32.3 RNG functional description

32.3.1 RNG block diagram

Figure 294 shows the RNG block diagram.

Figure 294. RNG block diagram



32.3.2 RNG internal signals

Table 310 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 310. RNG internal input/output signals

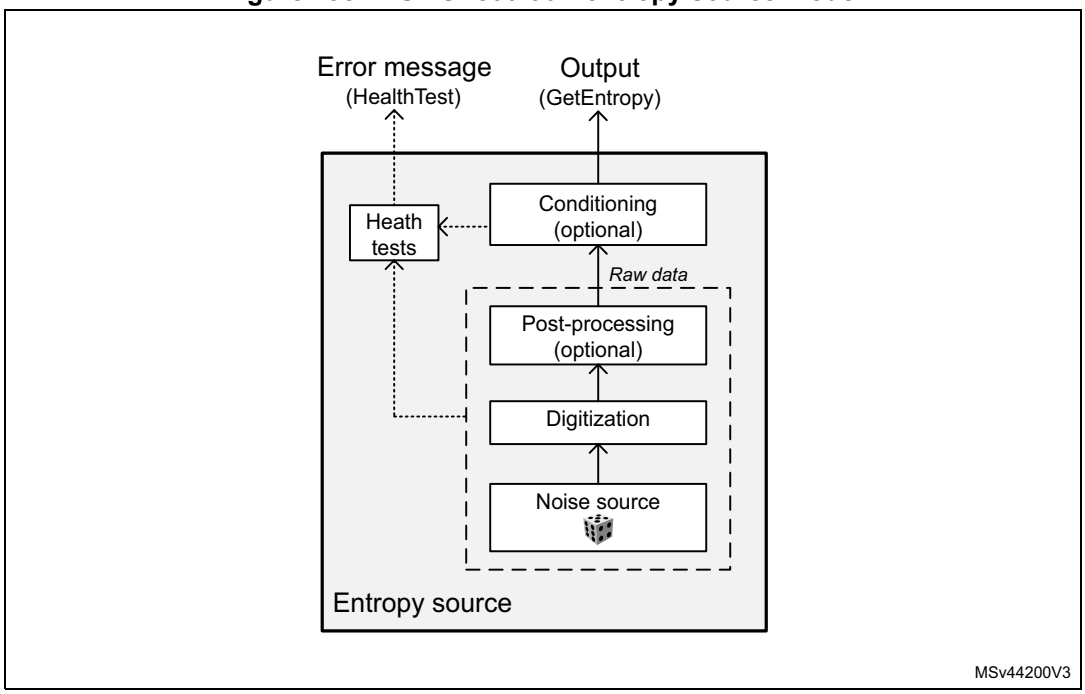
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	digital output	RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again.

32.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on *Figure 295*. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 295. NIST SP800-90B entropy source model



The components pictured above are detailed hereafter.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 32.3.8*. Multiple oscillators are also disabled for the configuration A (see *Table 313: RNG configurations*).
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In Section 32.6 the recommended RNG clock frequencies and associated divider value are given.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In non-NIST configuration B (as defined in *Section 32.6.2*) a normalization debiasing is applied,

that is half of the bits are taken from the sampled noise source, half of the bits are taken from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 32.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 32.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in *Section 32.6.2*).

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 711 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source:
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.
4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 32.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 32.3.7*.

Note: An interrupt can be generated when an error is detected.

Above the health test thresholds are modified by changing the value in the RNG_HTCR register. See Section 32.6 for details.

32.3.4 RNG initialization

The RNG simplified state machine is pictured on *Figure 296*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering startup behavior test on the raw data samples and known-answer tests.
3. When startup health tests are completed. During this time, three 128-bit noise source samples are used.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 32.5*.

Figure 296. RNG initialization overview

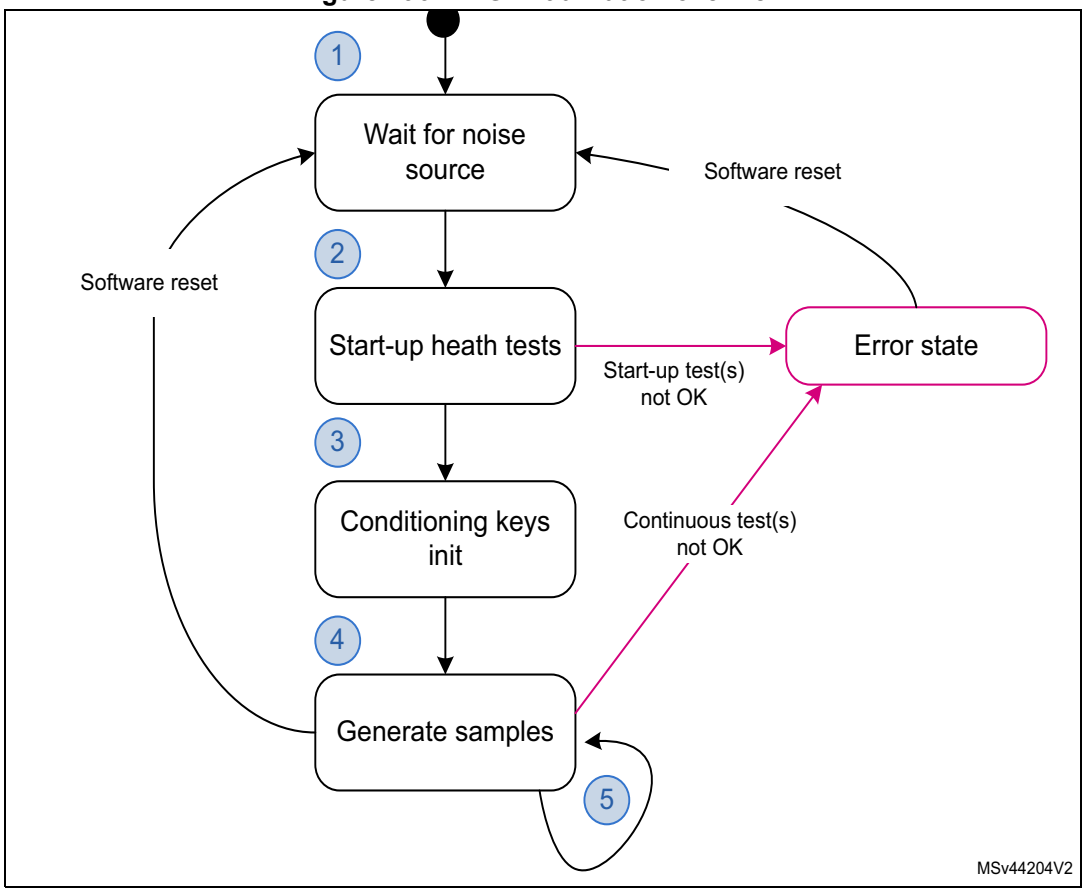


Figure 296 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes.

Note: When the RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in the RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

32.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check *Section 32.6* to verify if a specific RNG configuration is required for the application.
   – In this case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register

with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.

– If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   – No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   – A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   – If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 32.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Check *Section 32.6* to verify if a specific RNG configuration is required for the application.
   – If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   – If it is not the case only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   – No error occurred (the SEIS and CEIS bits must be set to 0)
   – A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 32.3.7* must be used.

Note: When data is not ready (DRDY = 0) RNG_DR returns zero.
It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration “B”, described in *Section 32.6*. The gain in random number generation speed is summarized in *Section 32.5*.

Low-power operation

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 32.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 32.3.3: Random number generation*.

32.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 32.6: RNG entropy source validation*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 32.3.1: RNG block diagram* for details (AHB and RNG clock domains).

32.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 32.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

Note: The clock error has no impact on generated random numbers that is the application can still read the RNG_DR register.

CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

When ARDIS is cleared (auto-reset enabled) clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.

When ARDIS is set (auto-reset disabled) the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

When the application sets the ARDIS bit in the RNG_CR register, the auto-reset is disabled. CONDRST must be used in step 1.

RNG tamper errors

When an unexpected error is found by the RNG an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral either using the central reset management or the global SoC reset. Then a proper initialization of the RNG is required, again.

32.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to the software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be reenabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round is done.
  Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 32.5: RNG processing time on page 1256*.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

When the application wants to disable the RNG clock it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and disabling the RNG kernel clock using the RCC.

Also, when application needs to enter a power mode where RNG is de-activated, it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and entering the low power mode using the PWR.

In the two cases above, to avoid unexpected consumption when RNG analog oscillators stay active, application can set the bit 13 in RNG_CR register. Setting this bit adds some marginal power consumption while RNGEN bit is set (RNG activated).

Note: The power modes where RNG is deactivated (that is retained or not available) can be found in the PWR section.

32.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 32.3.7: Error management*
- Clock error, see *Section 32.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 311*.

Table 311. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 32.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

32.5 RNG processing time

In recommended configuration A or C described in *Table 313*, the time between two sets of four 32-bit data is either:

- 203 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

With $f_{threshold} = 1.6 \times f_{RNG}$, for instance 77 MHz if f_{RNG}= 48 MHz. Value N is defined in *Section 32.6: RNG entropy source validation*.

Note: When CLKDIV is different from zero, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 203 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold} = 6.5 \times f_{RNG}$.

Initialization time

More time is needed for the first set of random numbers after the device exits reset (see Section 1.3.4: RNG initialization). Table below gives details on how to compute the time spent in each initialization step.

Table 312. RNG initialization times

Initialization step	Configuration A or C, reset value	Configuraton B
Wait for noise source, then startup health tests	Max((wait_noise[1] + 11 + 1024 x CLKDIV) RNG_cycles, 13 x 203 AHB_cycles)	Max(16 + 1035 RNG_cycles, 13 x 203 AHB_cycles)
Conditioning keys initialization	Max((1+2xN) x 128 x CLKDIV) RNG_cycles + 203 AHB_cycles, (128 x CLKDIV) RNG_cycles + (2xN+2) x 203 AHB_cycles)	Max (3 x 32 RNG_cycles + 203 AHB_cycles, 32 RNG_cycles + 4 x 203 AHB_cycles)

1. 192 RNG_cycles (configuration A or C), 16 RNG_cycles (reset value)

As an example, if AHB clock= 250 MHz and RNG clock= 48 MHz then the initialization time is around 40 µs in the configuration C (CLKDIV=1, N=2) and 26 µs in the configuration B.

32.6 RNG entropy source validation

32.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

32.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk = 48 MHz
- RNG configurations are described in *Table 313*. Note that only configuration A can be certified NIST SP800-90B. Refer to *Table 314* to select the best configuration for the application.
- Configuration C can be used when configuration A is not flagged as certified.

Table 313. RNG configurations

Configuration	RNG_CR bits						Loop number (N)	RNG_HTCR register	RNG_NSCR register
	NISTC bit	RNG_CONFIG1 [5:0]	CLKDIV [3:0]	RNG_CONFIG2 [2:0][1]	RNG_CONFIG3 [3:0][2]	CED bit			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program*.								
B	1	0x18	0x0[3]	0x0	0x0	0	1	0x0000 AAC7[4]	default
C	0	0x0F		0x0	0xD	0	2		default

1. 0x1 value is recommended instead of 0x0 for RNG_CONFIG2[2:0], when RNG power consumption is critical. See the end of *Section 32.3.8* for details.
2. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4

3. The noise source sampling must be 48 MHz or less. Hence, if the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 32.7.1* for details.
4. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).

Table 314. Configuration selection

Section criteria	Config A	Config B	Config C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Good	Very good
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using AIS-31 test suite (T0 to T8).
2. When speed is not enough for application a NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*STM32 microcontrollers random number generation validation using NIST statistical test suite*" available from *www.st.com*.

32.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

32.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV[3:0]			
rs	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG Config lock

0: Writes to the RNG_NSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR, RNG_NSCR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

When CONDRST is set to 0 by the software, its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:20 **RNG_CONFIG1[5:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 32.6: RNG entropy source validation*.

Writing any bit of RNG_CONFIG1 is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing these bits is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Bit 13 can be set when RNG power consumption is critical. See *Section 32.3.8: RNG low-power use*. Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 32.6: RNG entropy source validation* for recommended configuration.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset enabled

1: Auto-reset disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 32.3.7: Error management*.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection enabled

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing this bit is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

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

32.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)

1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequences has been detected:

- Runtime repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
- Startup or continuous adaptive proportion test on noise source failed.
- Startup post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.

32.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

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

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.

When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

32.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0003 FFFF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC6[2:1]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_ OSC6[0]	EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 17:15 **EN_OSC6[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 6. The bitfield has no effect otherwise.

Bits 14:12 **EN_OSC5[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 5. The bitfield has no effect otherwise.

Bits 11:9 **EN_OSC4[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 4. The bitfield has no effect otherwise.

Bits 8:6 **EN_OSC3[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

32.7.5 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x0000 72AC

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

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

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests. See *Section 32.6: RNG entropy source validation* for the recommended value.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

32.7.6 RNG register map

Table 315. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV [3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0					0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
	Reset value																										0	0			0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC6[2:0]			EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
	Reset value															1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

33 AES hardware accelerator (AES)

This section only applies to STM32H533/573xx products.

33.1 AES introduction

The AES hardware accelerator (AES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

AES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits.

AES has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required).

33.2 AES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 51 or 75 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data

- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if AES needs to process another message with a higher priority, then resume the original message

33.3 AES implementation

The devices have one AES peripheral, implemented as per the following table. It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 316. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

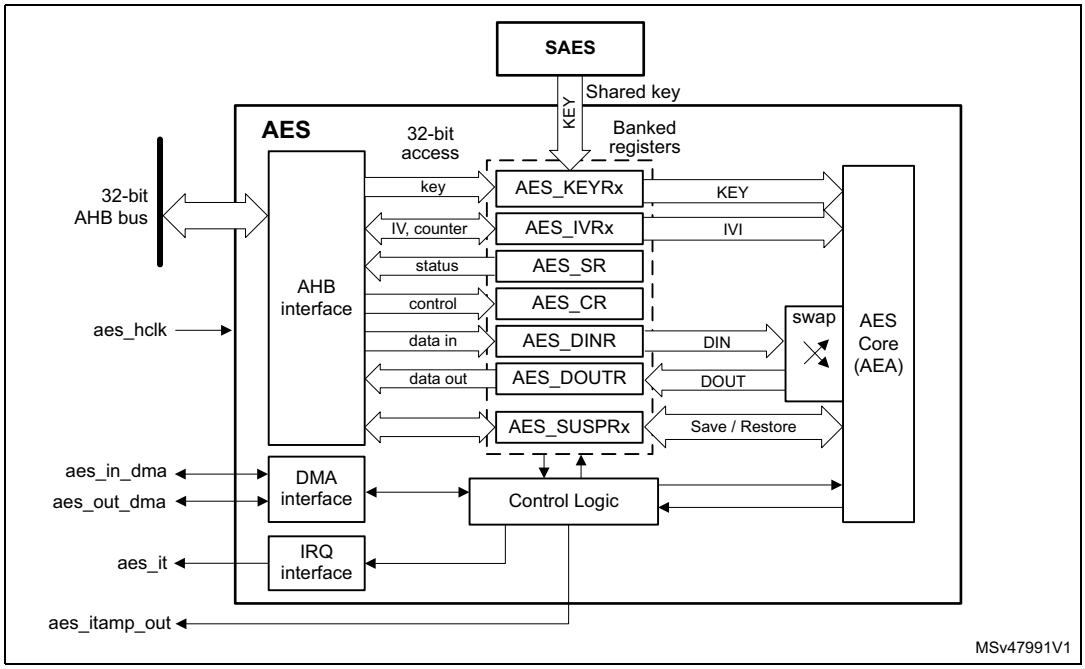
1. X = supported.

33.4 AES functional description

33.4.1 AES block diagram

Figure 297 shows the block diagram of AES.

Figure 297. AES block diagram



33.4.2 AES internal signals

Table 317 describes the user relevant internal signals interfacing the AES peripheral.

Table 317. AES internal input/output signals

Signal name	Signal type	Description
aes_hclk	Input	AHB bus clock
aes_it	Output	AES interrupt request
aes_in_dma	Input/Output	AES incoming data DMA single request/acknowledge
aes_out_dma	Input/Output	AES processed data DMA single request/acknowledge
aes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, AES automatically clears key registers. A reset is required for AES to be usable again.

33.4.3 AES reset and clocks

The AES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit controlled through the RCC.

33.4.4 AES symmetric cipher implementation

The AES hardware accelerator (AES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 318*. Those functions can be certified NIST PUB 140-3.

Table 318. AES approved symmetric key functions

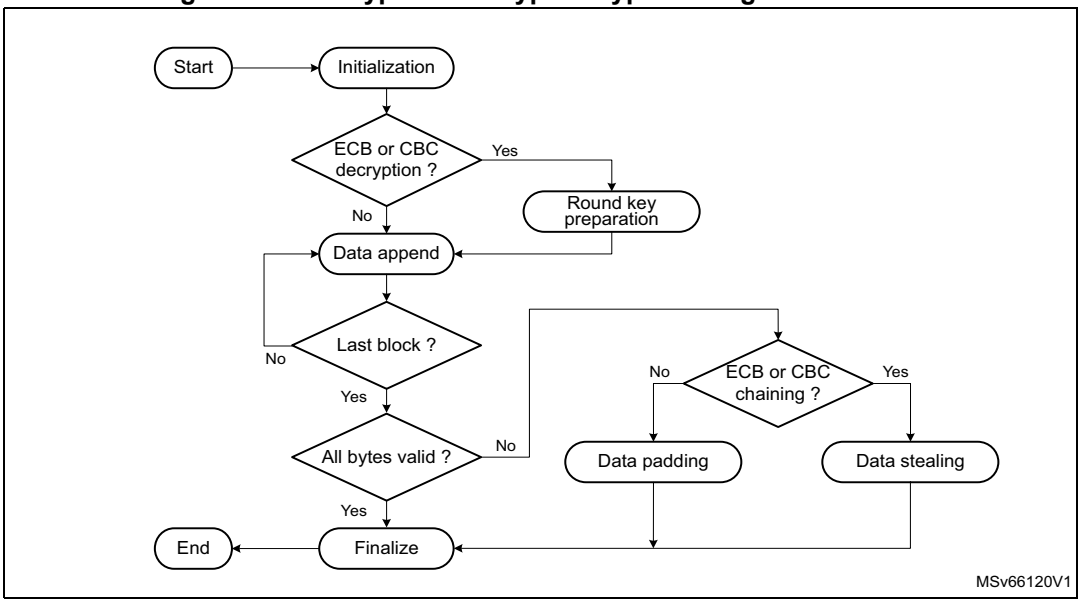
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

AES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any AES processing, following the sequence described in *Section 33.4.8*.

33.4.5 AES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 298. Encryption/ decryption typical usage



Initialization

The AES peripheral is initialized according to the chaining mode. Refer to *Section 33.4.9: AES basic chaining modes (ECB, CBC)* and *Section 33.4.10: AES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 33.4.7: AES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the AES peripheral when KEYVALID is set, by setting the EN bit of the AES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the AES_DINR register.
   b) Wait until the status flag CCF is set in the AES_ISR register, then read the four data words from the AES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).
3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the AES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the AES_DINR register, to allow sending the key to the AES processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the AES peripheral to control the data append, through the following sequence:

1. Enable interrupts from AES, by setting the CCFIE bit of the AES_IER register.
2. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the AES_DINR register.
4. Handle the data in the AES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the AES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the AES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the AES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the AES_DINR register and quit the interrupt service routine.

Note: AES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two AES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and AES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the AES peripheral input and the processed data from the AES peripheral output to the memory, as described in *Section 33.6: AES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for AES to compute a correct tag.
3. Enable the AES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the AES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 33.4.8* is not supported.

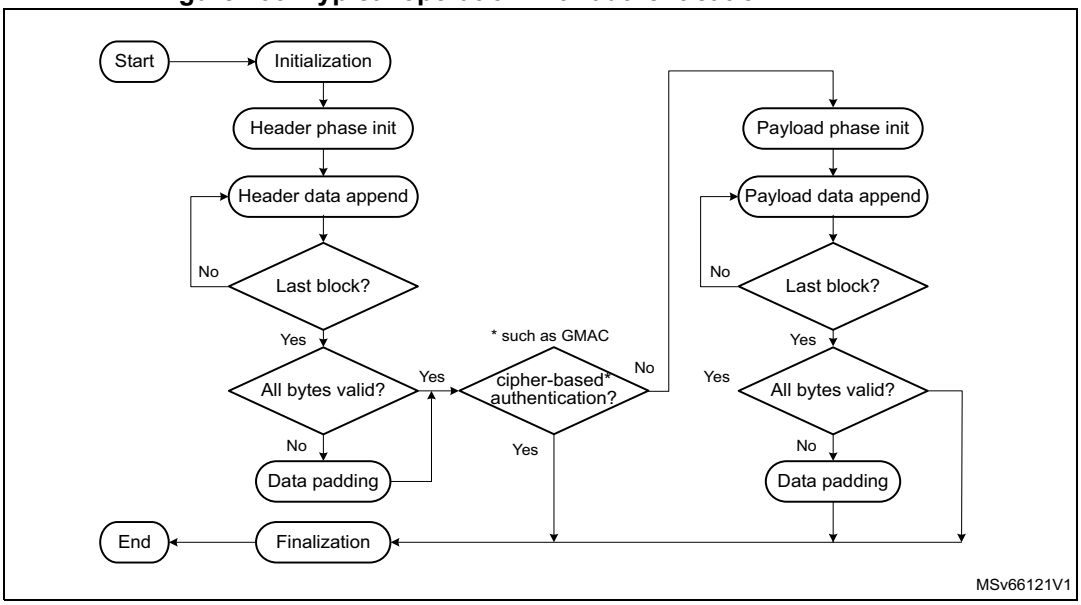
Note: The CCF flag has no use with this method because the reading of the AES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

33.4.6 AES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 299. Typical operation with authentication



Section 33.4.11: AES Galois/counter mode (GCM) and *Section 33.4.13: AES counter with CBC-MAC (CCM)* describe detailed sequences supported by AES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by AES is described in *Section 33.4.12: AES Galois message authentication code (GMAC)*.

33.4.7 AES ciphertext stealing and data padding

When using AES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since AES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since AES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

33.4.8 AES suspend and resume operations

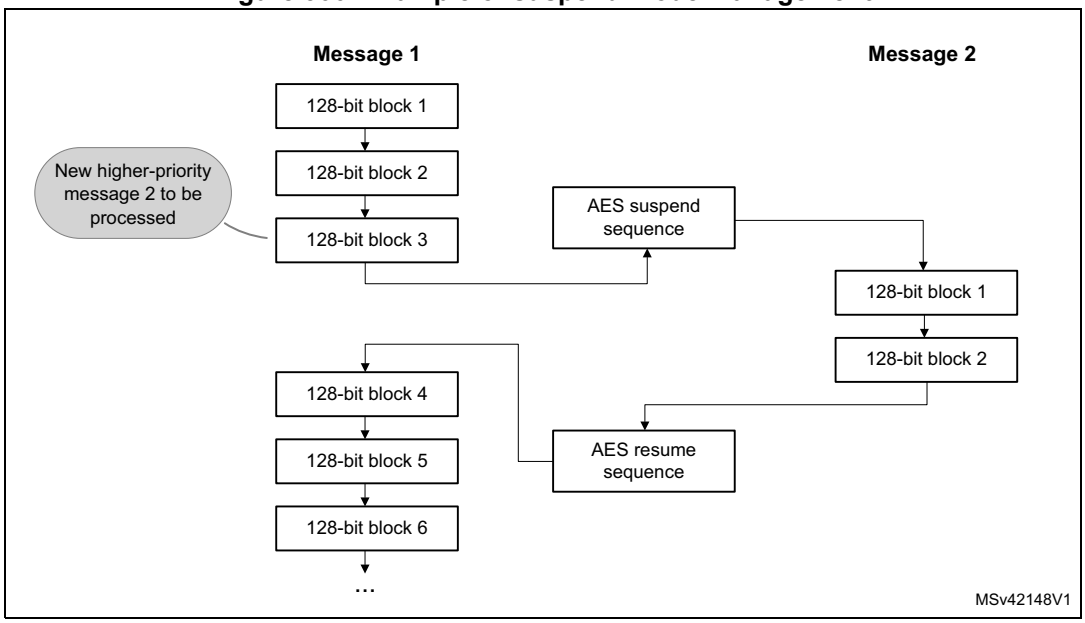
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as AES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

Figure 300 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 300. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

33.4.9 AES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 301 and *Figure 302* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 301. ECB encryption

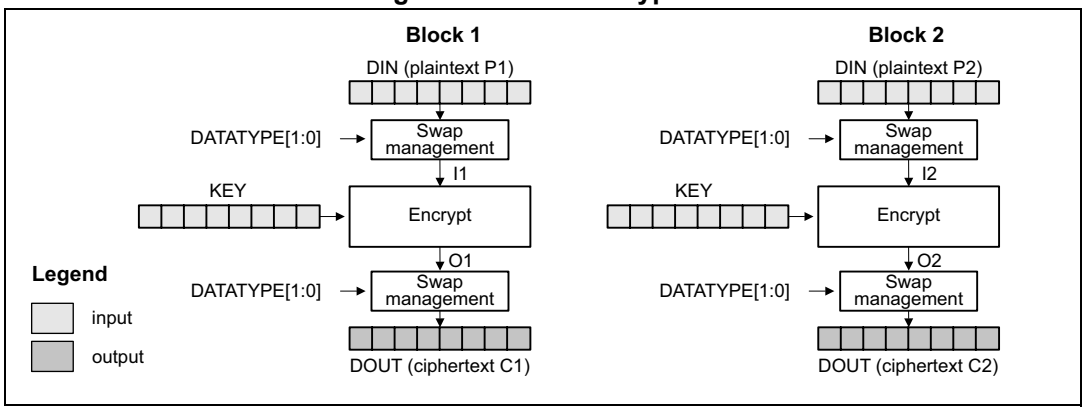
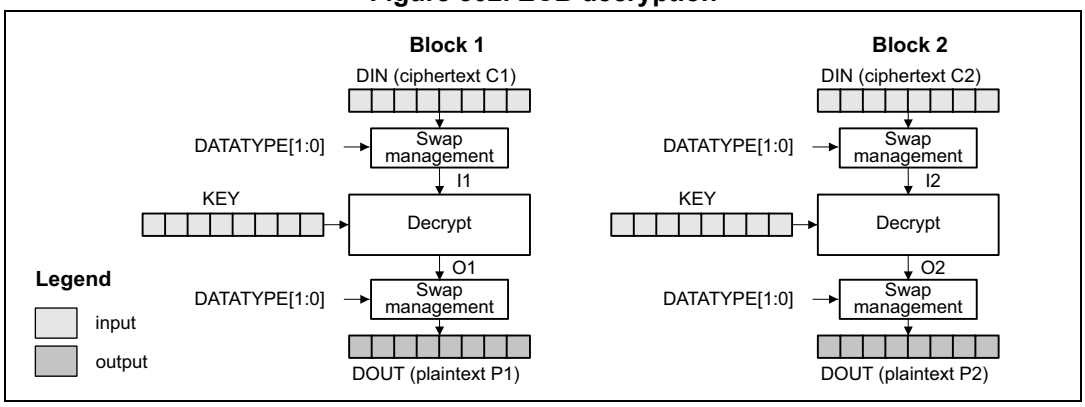


Figure 302. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 303 and *Figure 304* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 303. CBC encryption

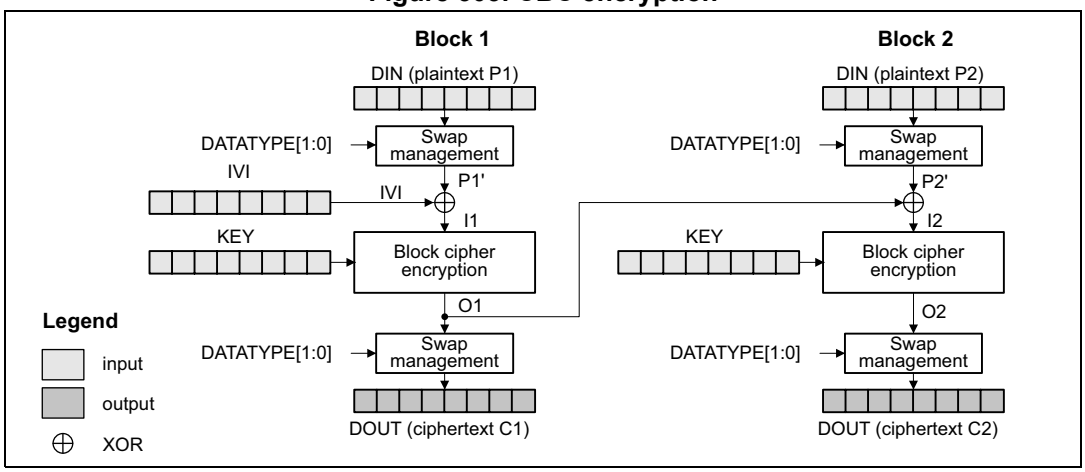
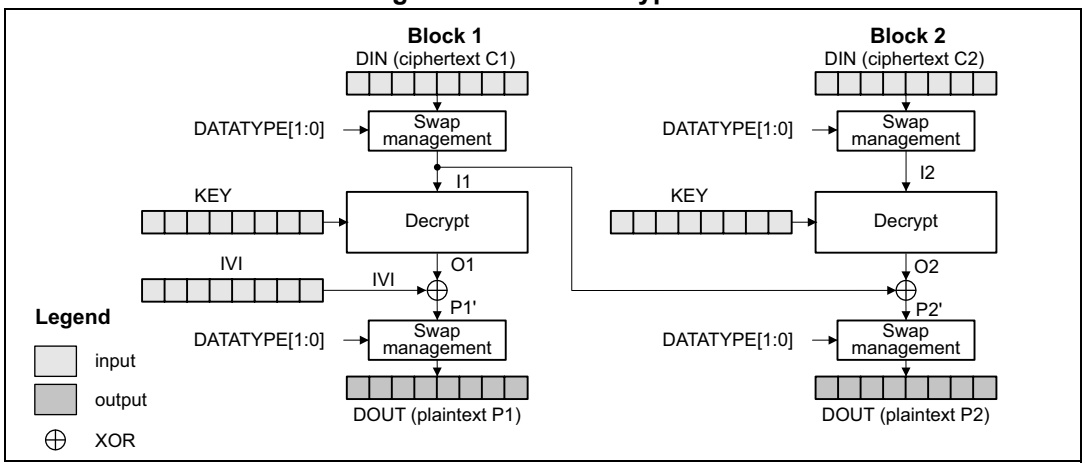


Figure 304. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 33.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 33.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.
7. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 33.4.7*.
   b) Append the cleartext block into AES as described in *Section 33.4.5*, then read the AES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 33.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 33.4.14*).
4. Wait until KEYVALID is set (the key loading completed).
5. Enable the AES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
6. Wait until the CCF flag in the AES_ISR register is set.
7. Clear the CCF flag, by setting the CCF bit of the AES_ICR register. The decryption key is available in the AES core and AES is disabled automatically.
8. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
9. Write the initialization vector into the AES_IVRx registers if CBC mode is selected in the previous step.
10. Enable the AES peripheral, by setting EN.
11. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 33.4.7*.
    b) Append the ciphertext block into AES as described in *Section 33.4.5*, then read the AES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
12. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. Read four times the AES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the AES_IVRx registers are altered during the data processing).

5. Disable the AES peripheral, by clearing EN.
6. Save the AES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Restore the AES_CR register (with correct KEYSIZE) then restore the AES_KEYRx registers. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 33.4.14*).
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
4. Restore the AES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the AES peripheral, by setting EN.

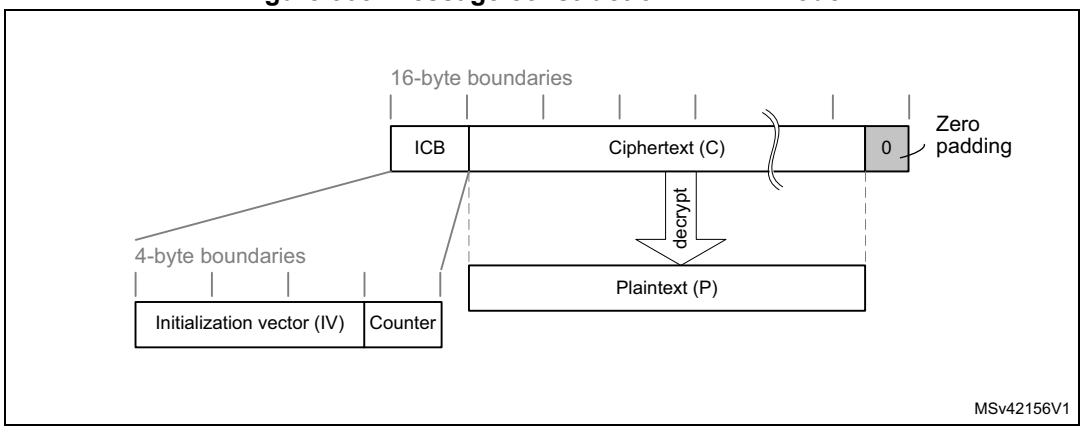
Note: It is not required to save the key registers as the application knows the original key.

33.4.10 AES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 305*.

Figure 305. Message construction in CTR mode



The structure of this message is:

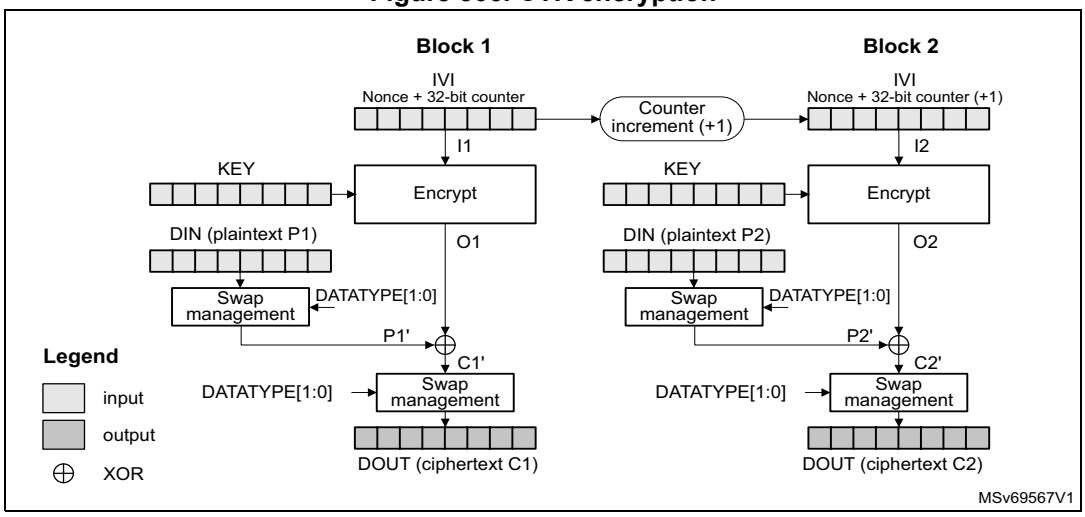
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 306 describes the counter (CTR) chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 306. CTR encryption



Initialization vectors in AES must be initialized as shown in *Table 319*.

Table 319. Counter mode initialization vector definition

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 33.4.5*, with the following sequence of events:

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
3. Write the initialization vector into the AES_IVRx registers according to *Table 319*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 33.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the AES peripheral, by setting EN.

7. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into AES as described in *Section 33.4.5*, then read the AES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
8. Finalize the sequence: disable the AES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 33.4.9: AES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

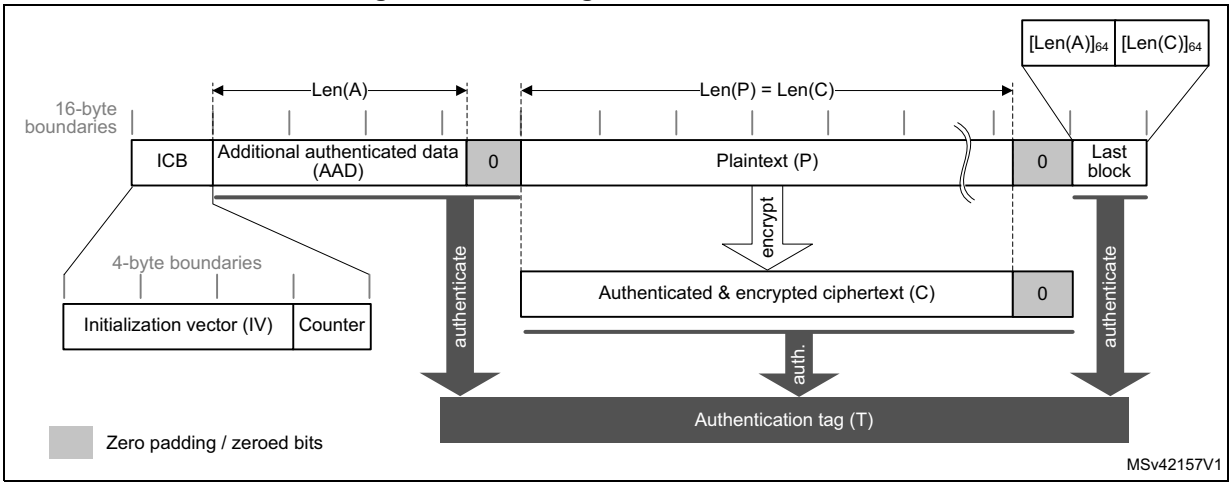
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

33.4.11 AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 307. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 321*.

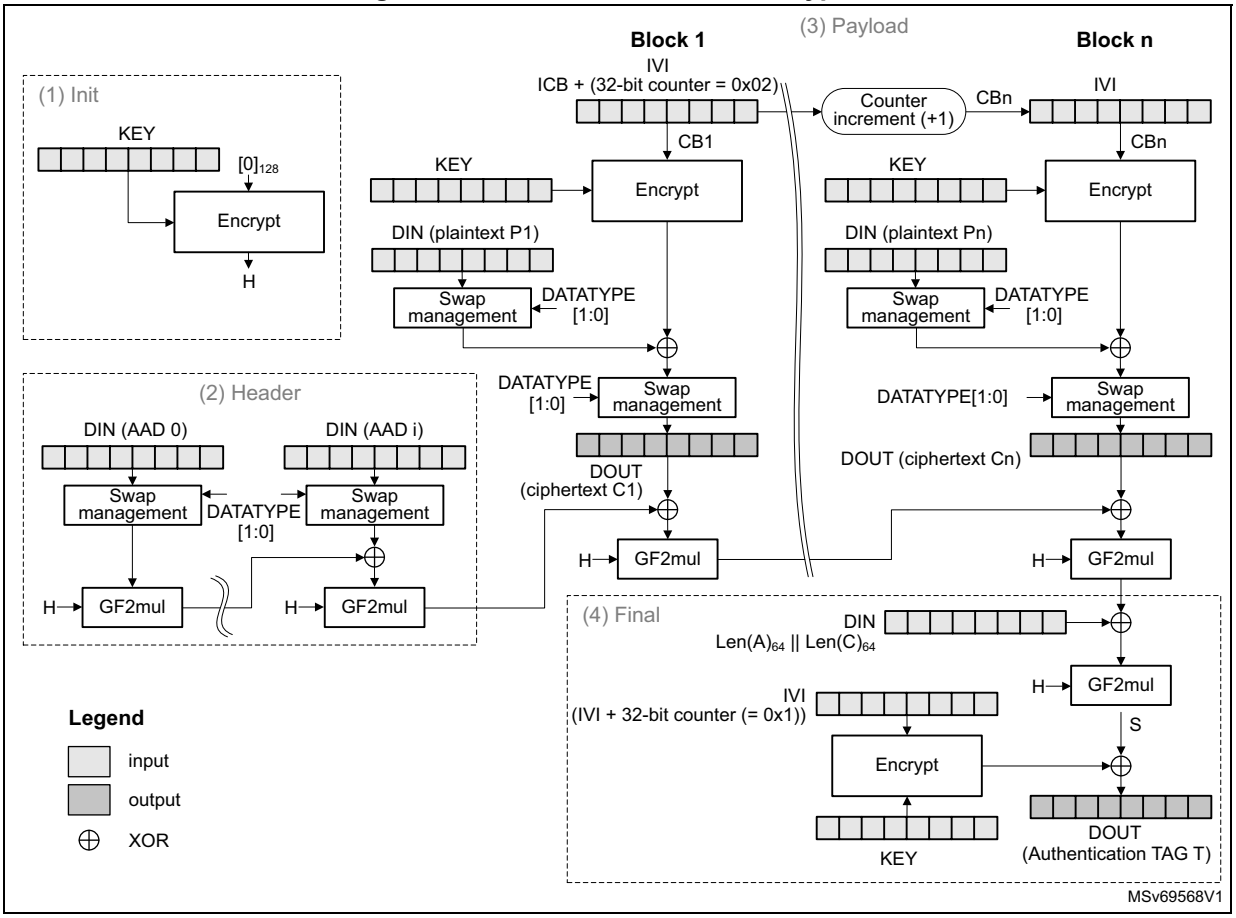
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 308 describes the GCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 308. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 320*.

Table 320. Initialization of IV registers in GCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 321*.

Table 321. GCM last block definition

Word order to AES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 33.4.6*, with the following sequence of events:

GCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the initialization vector in AES_IVRx registers according to *Table 320*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 33.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
7. Wait until the CCF flag is set in the AES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
8. Clear the CCF flag by setting the CCF bit of the AES_ICR register.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 33.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

11. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the AES peripheral by setting EN.

12. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 33.4.5*, then read the AES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

13. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
14. Write the final GCM block into the AES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 321*.
15. Wait until the CCF flag in the AES_ISR register is set.
16. Get the GCM authentication tag, by reading the AES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
18. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to AES_DINR normally (no swapping), while swapping is applied to tag data read from AES_DOUTR.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag of the AES_ISR register, by setting the CCF bit of the AES_ICR register. When GCM encryption payload phase is selected, verify that BUSY is cleared in AES_SR, to ensure that GF2mul hash function is completed.
4. Save the AES_SUSPRx registers in the memory.
5. In the payload phase, save the AES_IVRx registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
6. Disable the AES peripheral, by clearing EN.
7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

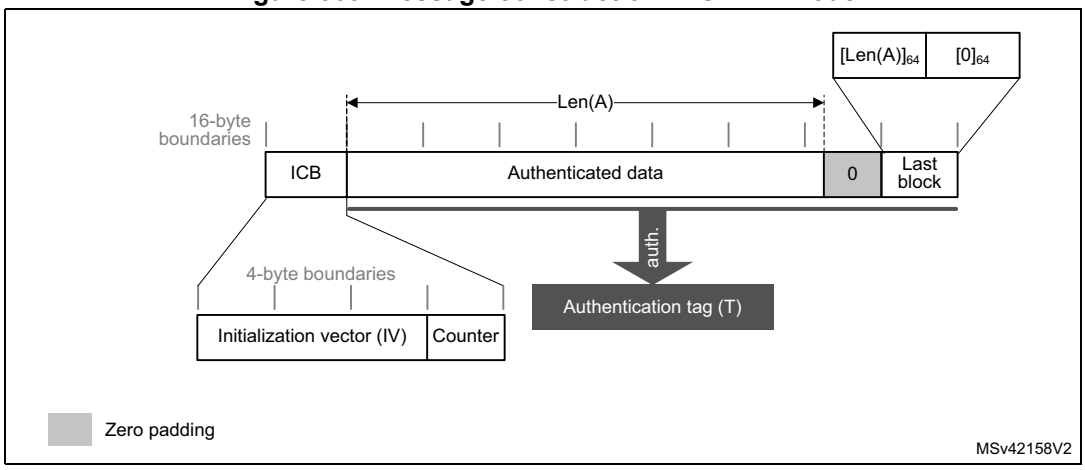
1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding AES_IVRx registers. In the header phase, write initial setting values back into the AES_IVRx registers.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 33.4.14*).
5. Enable the AES peripheral, by setting EN.

33.4.12 AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

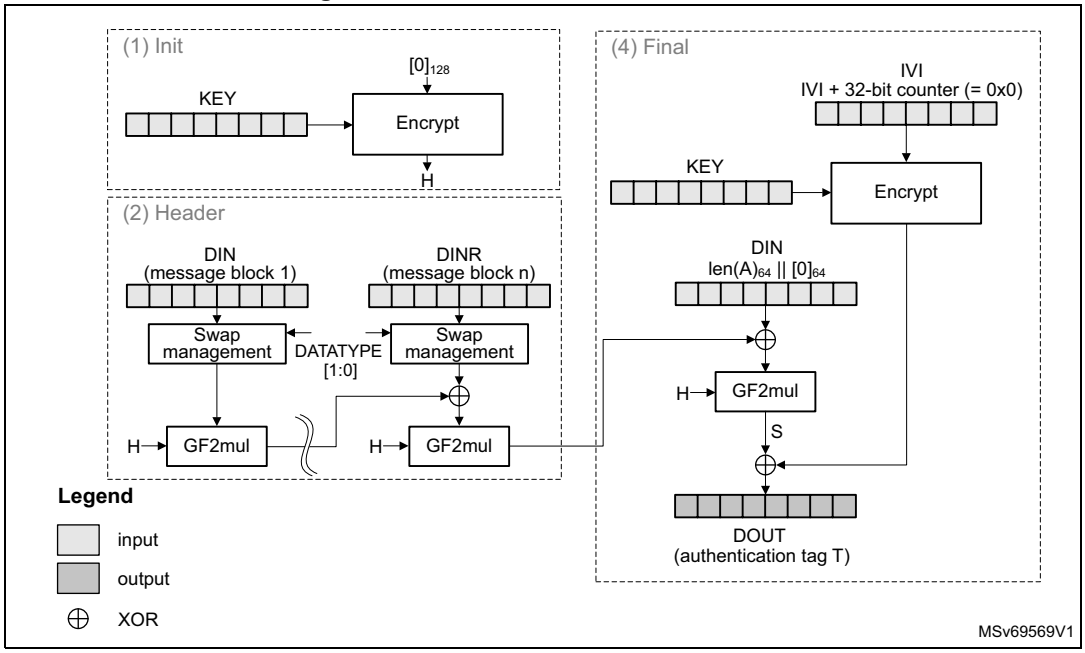
Figure 309. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 310 describes the GMAC chaining implementation in the AES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 310. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

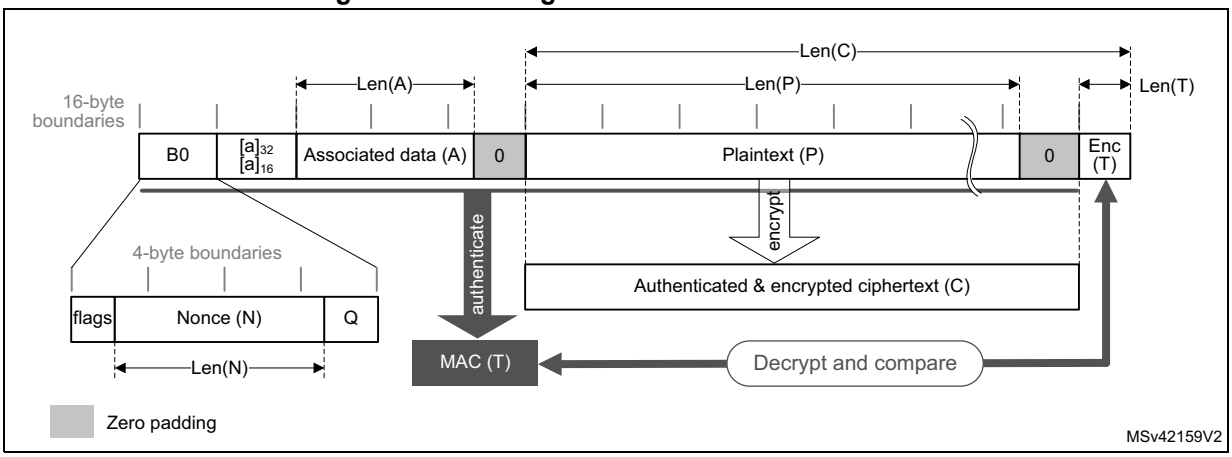
33.4.13 AES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 311. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 311*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 311*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

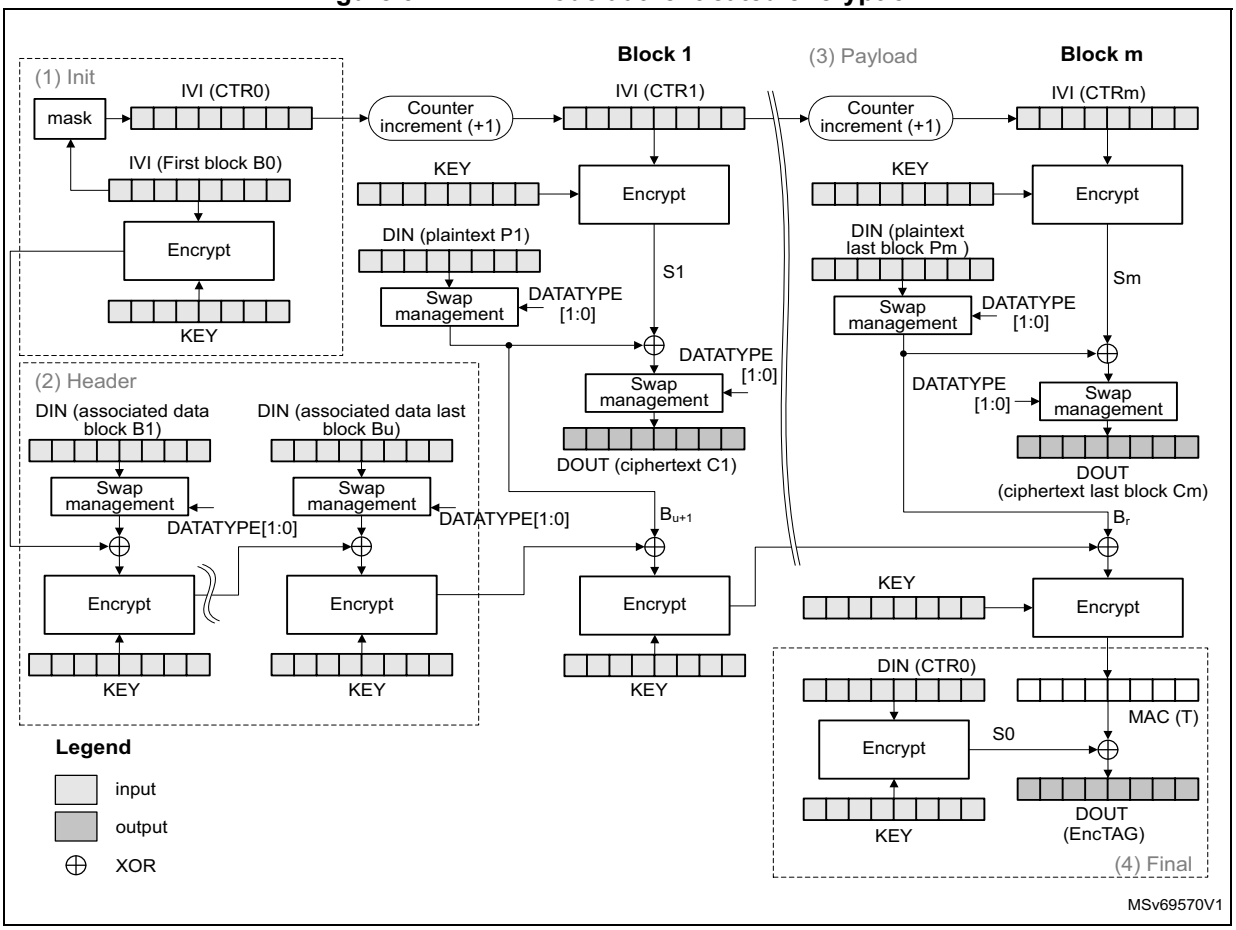
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 312 describes the CCM chaining implementation in the AES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 312. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 322*.

Table 322. Initialization of IV registers in CCM mode

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

AES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 33.4.6*, with the following sequence of events:

CCM initialize

1. Disable the AES peripheral, by clearing EN.
2. Initialize the AES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select the key mode, using KMOD[1:0]. If the key comes from the SAES peripheral, write KMOD[1:0] with 0x2, otherwise keep it at 0x0.
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
3. Write the B0 data in AES_IVRx registers according to *Table 322*.
4. Write the key into the AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key is transferred from the SAES peripheral (see *Section 33.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
7. Wait until the CCF flag in the AES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the AES peripheral, by setting EN.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 33.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

11. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the AES peripheral, by setting EN.
12. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into AES as described in *Section 33.4.5*, then read the AES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

13. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
14. Wait until CCF flag in the AES_ISR register is set.
15. Get the CCM authentication tag, by reading the AES_DOUTR register four times.
16. Clear the CCF flag, by setting the CCF bit of the AES_ICR register.
17. Disable the AES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from AES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the AES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when AES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the AES_ISR register is set (computation completed).
2. In the payload phase, read four times the AES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the AES_ISR register, by setting the CCF bit of the AES_ICR register.
4. Save the AES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the AES peripheral, by clearing EN.

7. Save the current AES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the AES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding AES_SUSPRx registers.
3. Restore AES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the AES_CR and AES_KEYRx registers if KMOD[1:0] is at 0x0. If KMOD[1:0] is at 0x2, the key must be transferred again from the SAES peripheral (see *Section 33.4.14*).
5. Enable the AES peripheral, by setting EN.

33.4.14 AES key sharing with secure AES co-processor

The AES peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral and when it is ready, the AES application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in the section *SAES operations with shared keys* in the *SAES* section of this document.

1. In SAES peripheral, the application encrypts (wraps) the key to share in Shared-key mode (KMOD[1:0] at 0x2).
2. Each time the shared key is required in AES peripheral, the application decrypts it in the SAES peripheral in Shared-key mode (KMOD[1:0] at 0x2).
3. Once the shared key is decrypted (unwrapped) and loaded in SAES_KEYRx registers it can be shared with AES. To load the shared key in AES, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transfered by hardware into the AES_KEYRx registers and the BUSY flag in the AES_SR register set.
4. Once the key transfer is completed, the BUSY flag is cleared and the KEYVALID flag set in the AES_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KEIF flag is set in the AES_ISR register, either the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is completed, the AES is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in AES peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

33.4.15 AES data registers and data swapping

Data input and output

A 16-byte data block enters the AES peripheral with four successive 32-bit word writes into the AES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the AES peripheral with four successive 32-bit word reads of the AES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 323* “no swapping” option for details.

Data swapping

The AES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the AES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the AES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both AES input and output.

Note: The data in AES key registers (AES_KEYRx) and initialization vector registers (AES_IVRx) are not sensitive to the swap mode selection.

The AES data swapping feature is summarized in *Table 323* and *Figure 313*.

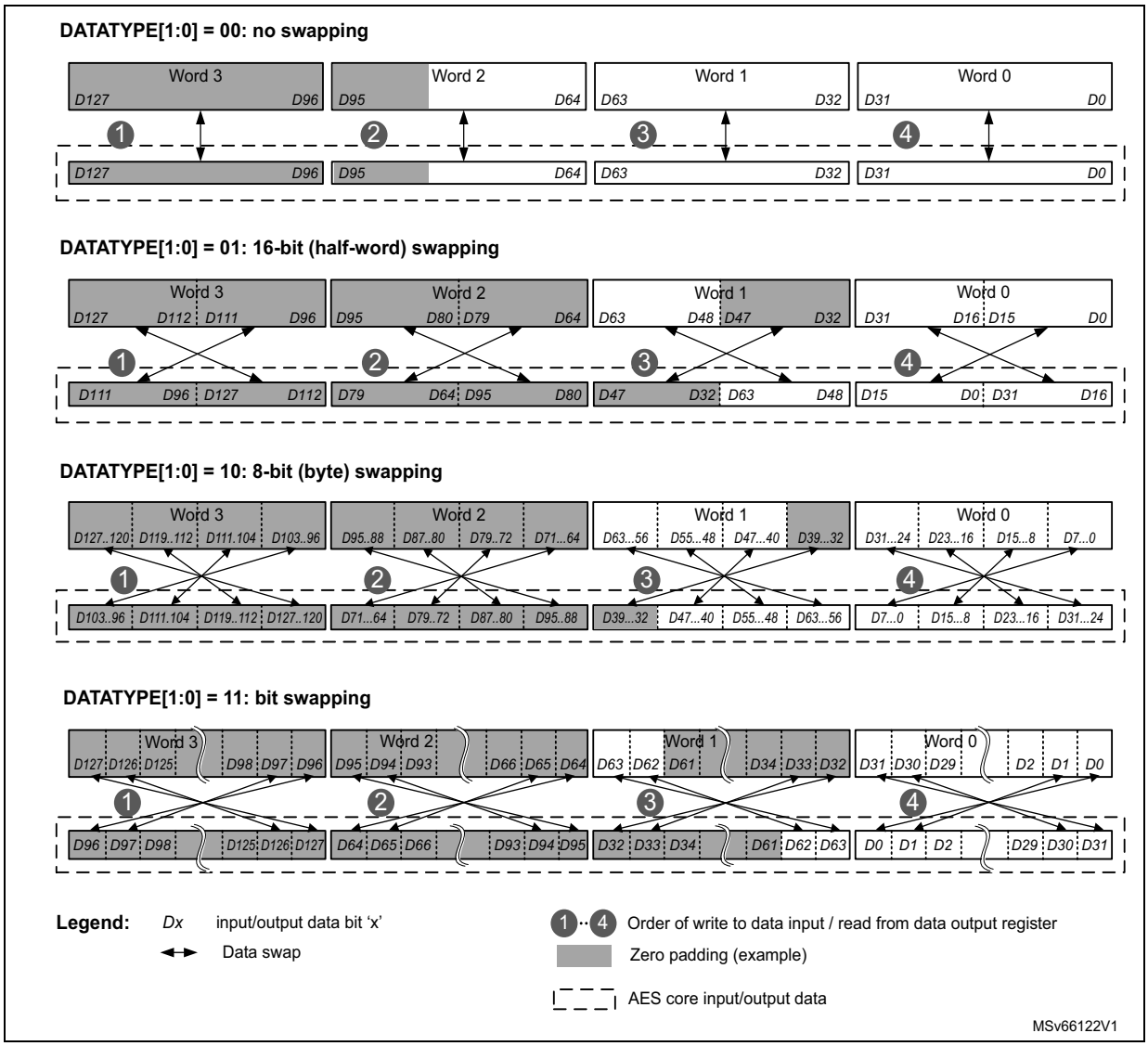
Table 323. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**

Table 323. AES data swapping example (continued)

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 313. 128-bit block construction according to the data type



Data padding

Figure 313 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

33.4.16 AES key registers

The eight AES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 324*. Reads are not allowed for security reason.

Note: In memory and in AES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 324. Key endianness in AES_KEYRx registers (128/256-bit keys)

AES_KEYR7 [31:0]	AES_KEYR6 [31:0]	AES_KEYR5 [31:0]	AES_KEYR4 [31:0]	AES_KEYR3 [31:0]	AES_KEYR2 [31:0]	AES_KEYR1 [31:0]	AES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the AES_KEYRx registers are ignored when AES peripheral is enabled (EN bit set). The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes:

- AES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- AES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 33.4.18*).

33.4.17 AES initialization vector registers

The four AES_IVRx registers store the initialization vector (IV) information, as shown in *Table 325*. They can only be written if the AES peripheral is disabled (EN cleared).

Note: In memory and in AES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 325. IVI bitfield spread over AES_IVRx registers

AES_IVR3[31:0]	AES_IVR2[31:0]	AES_IVR1[31:0]	AES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, AES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

33.4.18 AES error management

The AES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the AES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: AES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 33.5: AES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the AES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: AES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 33.5: AES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the AES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the AES_ISR register and clear the KEYVALID bit of the AES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 33.4.16: AES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.

- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 33.4.14: AES key sharing with secure AES co-processor* for details.

The KEIF flag is cleared with corresponding bit of the AES_ICR register. An interrupt is generated if the KEIE bit of the AES_IER register is set. For more details, refer to *Section 33.5: AES interrupts*.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring AES.

33.5 AES interrupts

There are multiple individual maskable interrupt sources generated by the AES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)

See *Section 33.4.18: AES error management* for details on AES errors.

These sources are combined into a common interrupt signal from the AES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable AES interrupt sources individually by setting/clearing the corresponding enable bit of the AES_IER register.

The status of the individual maskable interrupt sources can be read from the AES_ISR register. They are cleared by setting the corresponding bit of the AES_ICR register.

Table 326 gives a summary of the available features.

Table 326. AES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
AES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)

1. Bit of the AES_ICR register.
2. Flag of the AES_SR register, mirrored by the flag RWEIF of the AES_ISR register.

33.6 AES DMA requests

The AES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the AES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

AES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with AES can be found in *Appending data using DMA* subsection of *Section 33.4.5: AES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into AES. AES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the AES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 33.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from AES. AES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the AES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, AES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the AES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when AES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

33.7 AES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 327. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
128-bit	Encryption or decryption(1)	ECB, CBC, CTR	51
	Key preparation	-	59
256-bit	Encryption or decryption(1)	ECB, CBC, CTR	75
	Key preparation	-	82

1. Excluding key preparation time (ECB and CBC only).

Table 328. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Encryption/ Decryption	GCM	64	35	51	59
		CCM	63	55	114	58
256-bit	Encryption/ Decryption	GCM	88	35	75	75
		CCM	87	79	162	82

1. Data insertion can include wait states forced by AES on the AHB bus (maximum 3 cycles, typical 1 cycle).

33.8 AES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

33.8.1 AES control register (AES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]
rw						rw	rw	rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: AES peripheral software reset

Setting the bit resets the AES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the AES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the AES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable.

0x2: Shared key mode. If shared key mode is properly initialized in SAES peripheral, the AES peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in AES key registers when BUSY bit is cleared and KEYVALID is set in the AES_SR register. Key error flag KEIF is set otherwise in the AES_ISR register.

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE:** Key size selection
This bitfield defines the key length in bits of the key used by AES.
0: 128-bit
1: 256-bit
Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection
This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable
This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable
This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to AES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode
This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode
This bitfield selects the AES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the AES_DINR register or read from the AES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 33.4.15: AES data registers and data swapping*.

0x0: No swapping (32-bit data).

0x1: Half-word swapping (16-bit data)

0x2: Byte swapping (8-bit data)

0x3: Bit-level swapping

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the AES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the AES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

– The bit cannot be set as long as KEYVALID is cleared

33.8.2 AES status register (AES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

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

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

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in AES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KMOD[1:0] is different from zero, the BUSY flag is automatically set by AES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is usually set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the AES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 33.4.16: AES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether AES is idle or busy.
0: Idle
1: Busy
AES is flagged as idle when disabled (when EN is low) or when the last processing is completed.
AES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), or transferring a shared key from the SAES peripheral.
When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the AES_DINR register occurred. When set WRERRF bit has no impact on the AES operations.
0: No error
1: Unexpected write to AES_DINR register occurred during computation or data output phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the AES_DOUTR register occurred. When set RDERRF bit has no impact on the AES operations.
0: No error
1: Unexpected read to AES_DOUTR register occurred during computation or data input phase.
The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.
The flag is cleared by setting the RWEIF bit of the AES_ICR register.

Bit 0 Reserved, must be kept at reset value.

33.8.3 AES data input register (AES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the AES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.
Reads return zero.

33.8.4 AES data output register (AES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the AES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

33.8.5 AES key register 0 (AES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the AES_CR register.

Writes to AES_KEYRx registers are ignored when AES is enabled (EN bit set). When the key comes from the SAES peripheral (KMOD[1:0] at 0x2), writes to key registers are also ignored and they result in setting the KEIF bit of the AES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

33.8.6 AES key register 1 (AES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.7 AES key register 2 (AES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.8 AES key register 3 (AES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.9 AES initialization vector register 0 (AES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

AES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in AES_CR register

33.8.10 AES initialization vector register 1 (AES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

33.8.11 AES initialization vector register 2 (AES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

33.8.12 AES initialization vector register 3 (AES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the AES_IVR0 register for description of the IVI[128:0] bitfield.

33.8.13 AES key register 4 (AES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.14 AES key register 5 (AES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.15 AES key register 6 (AES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.16 AES key register 7 (AES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the AES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing AES_KEYRx registers.

33.8.17 AES suspend registers (AES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **SUSP[31:0]**: Suspend data

AES_SUSPRx registers contain the complete internal register states of the AES when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 33.4.8: AES suspend and resume operations* for more details.

Clearing EN bit of the AES_CR register clears this register to zero.

AES_SUSPRx registers are not used in other chaining modes than GCM, GMAC or CCM.

33.8.18 AES interrupt enable register (AES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

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

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

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the AES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the AES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

33.8.19 AES interrupt status register (AES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

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

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

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

The flag setting generates an interrupt if the KEIE bit of the AES_IER register is set. It also clears the key registers and the KEYVALID flag in the AES_SR register.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

KEIF is raised upon any of the following events:

–AES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 register, or reverse. For KEYSIZE set, AES_KEYR0 then AES_KEYR1 then AES_KEYR2 then AES_KEYR3 then AES_KEYR4 then AES_KEYR5 then AES_KEYR6 then AES_KEYR7, or reverse).

–AES fails to load the key shared by the SAES peripheral (KMOD[1:0] = 0x2).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the AES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the AES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the AES_IER register is set.

The flag is cleared by setting the corresponding bit of the AES_ICR register.

33.8.20 AES interrupt clear register (AES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

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

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

Bit 2 **KEIF:** Key error interrupt flag clear
Setting this bit clears the KEIF status bit of the AES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear
Setting this bit clears the RWEIF status bit of the AES_ISR register, and clears both RDERRF and WRERRF flags in the AES_SR register.

Bit 0 **CCF:** Computation complete flag clear
Setting this bit clears the CCF status bit of the AES_ISR register.

33.8.21 AES register map

Table 329. AES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	AES_CR	IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1]	KMOD[0]	NPBLB[3:0]				Res.	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0						0	0	0	0	0	0		0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	AES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	AES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	AES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	AES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	AES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 329. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	AES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	AES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	AES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	AES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	AES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	AES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	AES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	AES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	AES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	AES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	AES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	AES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	AES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	AES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	AES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	AES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	AES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	AES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Table 329. AES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	AES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIE	RWEIE	CCFIE
	Reset value																														0	0	0
0x304	AES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x308	AES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEIF	RWEIF	CCF
	Reset value																														0	0	0
0x309-0x3FF	Reserved	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

34 Secure AES coprocessor (SAES)

This section only applies to STM32H533/573xx products.

34.1 SAES introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/ decryption (wrapped-key mode) and key sharing with faster AES peripheral (shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the AES peripheral.

34.2 SAES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), certified SESIP and PSA security assurance level 3
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster AES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together

- Security context enforcement for keys
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if SAES needs to process another message with a higher priority, then resume the original message

34.3 SAES implementation

The devices have one SAES peripheral, implemented as per the following table. It shares the key with the AES peripheral. For comparison, the AES peripheral is also included in the table.

Table 330. AES versus SAES features

Modes or features(1)	AES	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	51	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and AES	X	
Key sizes in bits	128, 256	128, 256

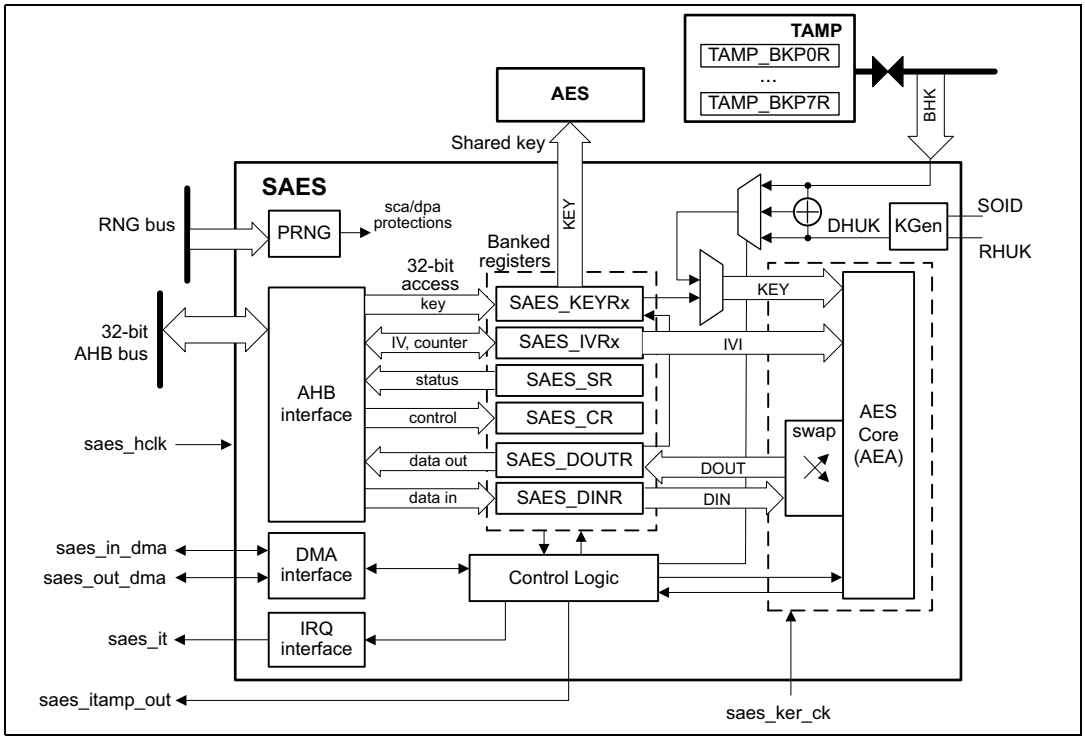
1. X = supported.

34.4 SAES functional description

34.4.1 SAES block diagram

Figure 314 shows the block diagram of SAES.

Figure 314. SAES block diagram



Note: AES represents the AES peripheral.

34.4.2 SAES internal signals

Table 331 describes the user relevant internal signals interfacing the SAES peripheral.

Table 331. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_ker_ck	Input	SAES kernel clock.
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	SAES incoming data DMA single request/acknowledge
saes_out_dma	Input/Output	SAES processed data DMA single request/acknowledge
saes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.
RHUK	Input	256-bit root hardware unique key (non-volatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)

Table 331. SAES internal input/output signals (continued)

Signal name	Signal type	Description
BHK(1)	Input	256-bit boot hardware key (BHK) stored in tamper-resistant secure backup registers and written by a secure code during boot. Once written, this key cannot be read nor written by any application until the next product reset.
SOID	Input	Static operating ID hardware input used during each DHUK computation. Its change causes the erasure of key registers and sets the KEIF flag.

1. Connected to a set of backup registers in TAMP peripheral that are written, then read/write locked, by the application software (see *Section 34.4.17* for details).

34.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit and a dedicated kernel clock, controlled through the RCC.

The SAES reset in RCC is mandatory when a hardware tamper event occurs in SAES. Refer to *Managing tamper errors* in *Section 34.4.19* for details.

After clocking then releasing the reset of the SAES peripheral in the RCC, SAES automatically draws random numbers from the RNG, setting the BUSY bit of the SAES_SR register. Refer to the *RNG error interrupt flag (RNGEIF)* in *Section 34.4.19* for details.

This background task must be completed before the device enters a low-power mode.

34.4.4 SAES symmetric cipher implementation

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 332*. Those functions can be certified NIST PUB 140-3.

Table 332. SAES approved symmetric key functions

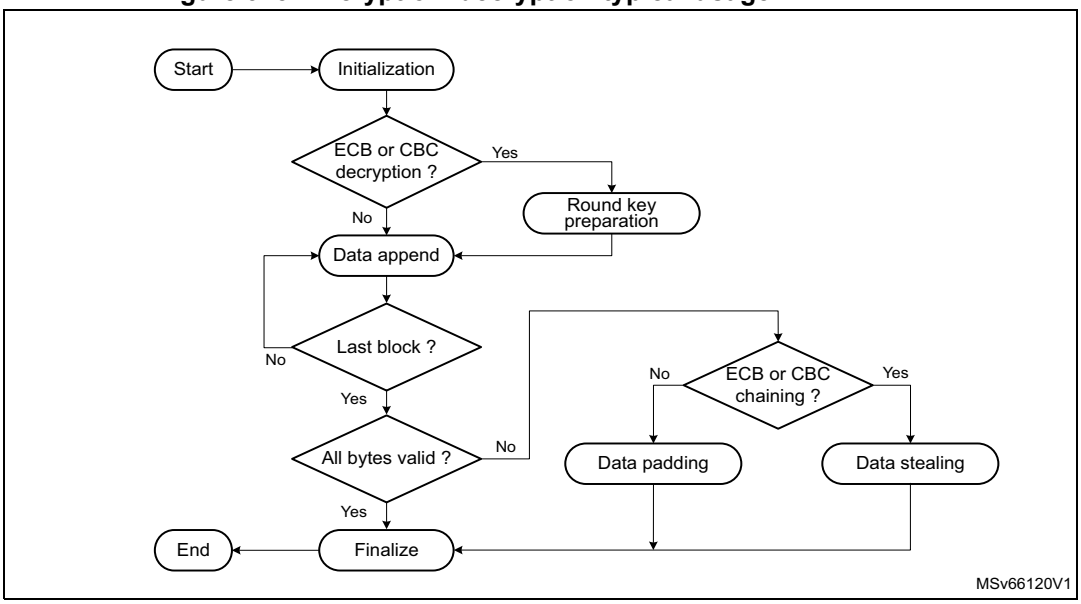
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any SAES processing, following the sequence described in *Section 34.4.8*.

34.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 315. Encryption/ decryption typical usage



Initialization

The SAES peripheral is initialized according to the chaining mode. Refer to *Section 34.4.9: SAES basic chaining modes (ECB, CBC)* and *Section 34.4.10: SAES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 34.4.7: SAES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral when KEYVALID is set, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 34.6: SAES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 34.4.8* is not supported.

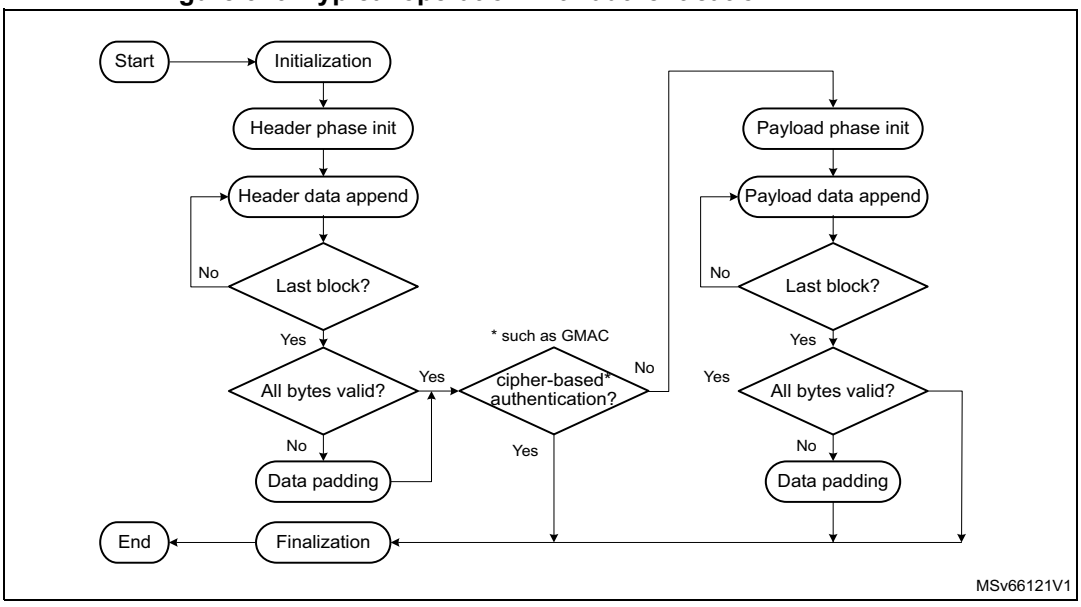
Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

34.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 316. Typical operation with authentication



Section 34.4.11: SAES Galois/counter mode (GCM) and *Section 34.4.13: SAES counter with CBC-MAC (CCM)* describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in *Section 34.4.12: SAES Galois message authentication code (GMAC)*.

34.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A*, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since SAES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since SAES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

34.4.8 SAES suspend and resume operations

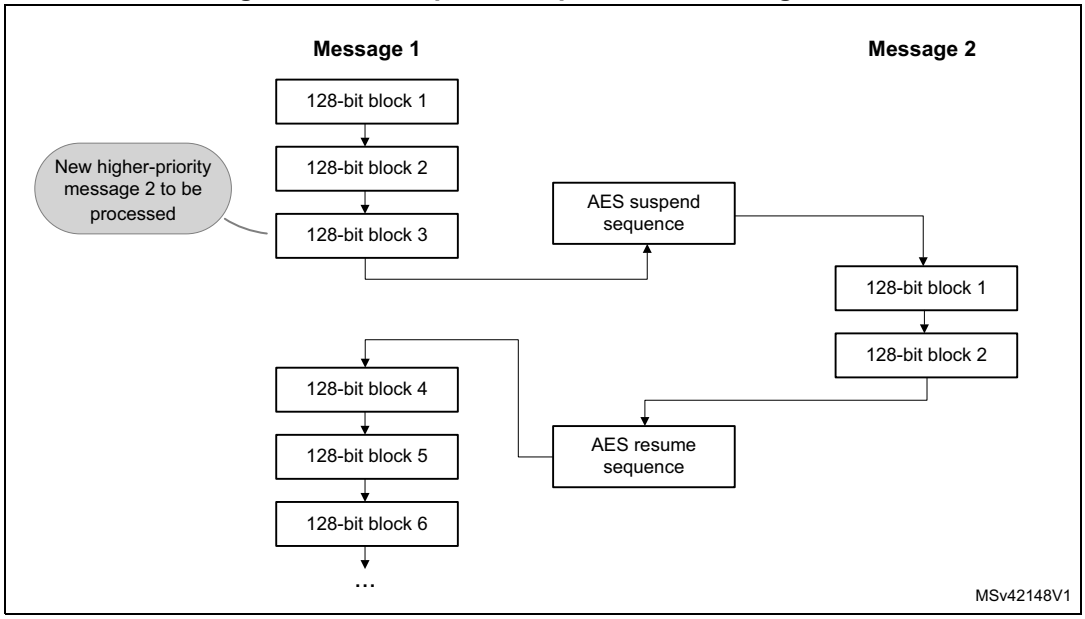
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

Figure 317 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 317. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

34.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 318 and *Figure 319* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 318. ECB encryption

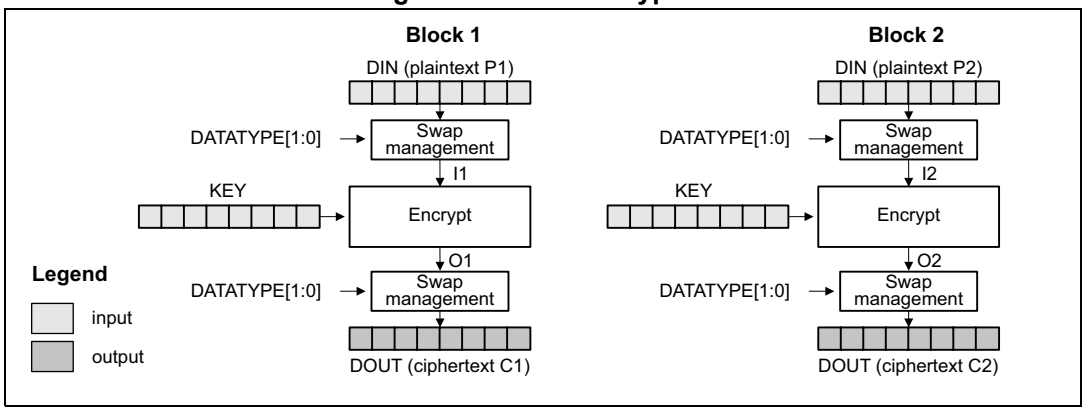
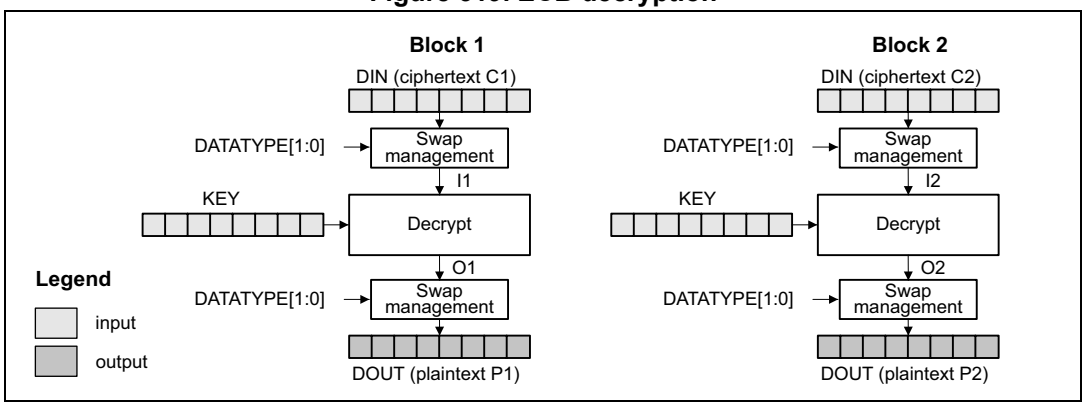


Figure 319. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 320 and *Figure 321* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 320. CBC encryption

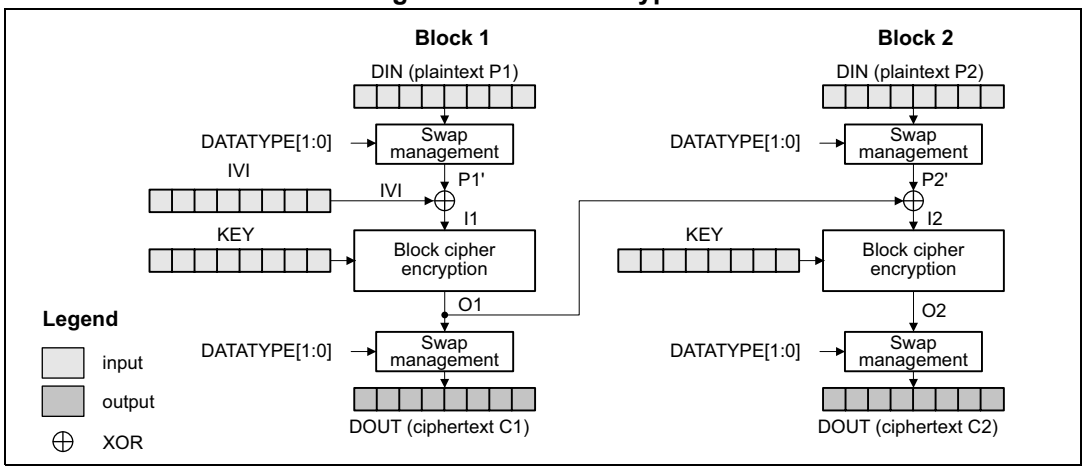
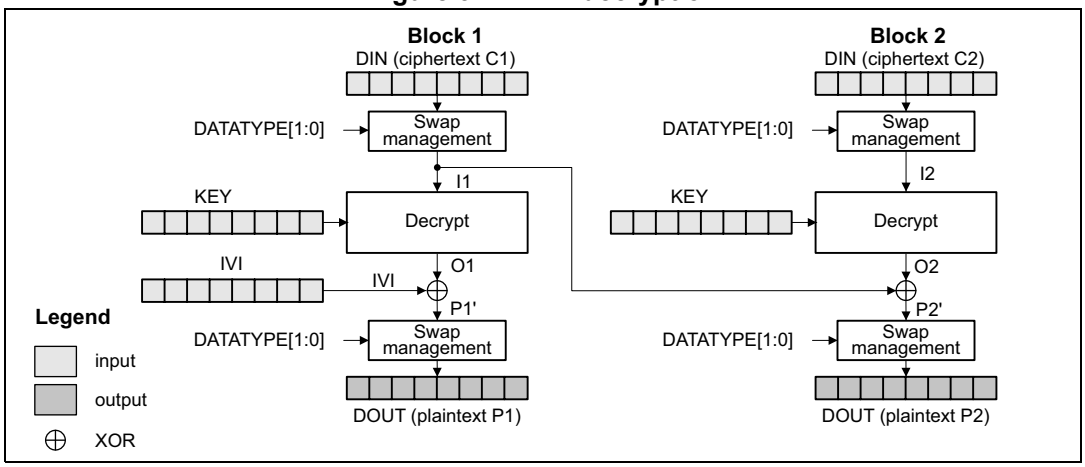


Figure 321. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 34.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 34.4.14* (wrapped keys) and *Section 34.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 34.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 34.4.7*.
   b) Append the cleartext block into SAES as described in *Section 34.4.5*, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 34.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 34.4.14* (wrapped keys) and *Section 34.4.15* (shared keys).

4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 34.4.17: SAES key registers* for details.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag in the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 34.4.7*.
    b) Append the ciphertext block into SAES as described in *Section 34.4.5*, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the SAES_ISR register is set (computation completed).
2. Read four times the SAES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).
5. Disable the SAES peripheral, by clearing EN.
6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from key registers, refer to *Section 34.4.17: SAES key registers* for details.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).

4. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the SAES peripheral, by setting EN.

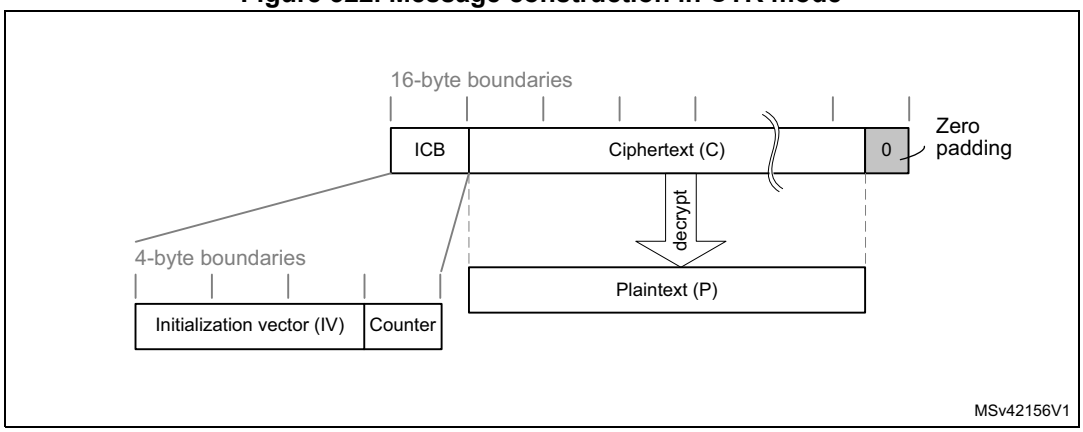
Note: It is not required to save the key registers as the application knows the original key.

34.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 322*.

Figure 322. Message construction in CTR mode



The structure of this message is:

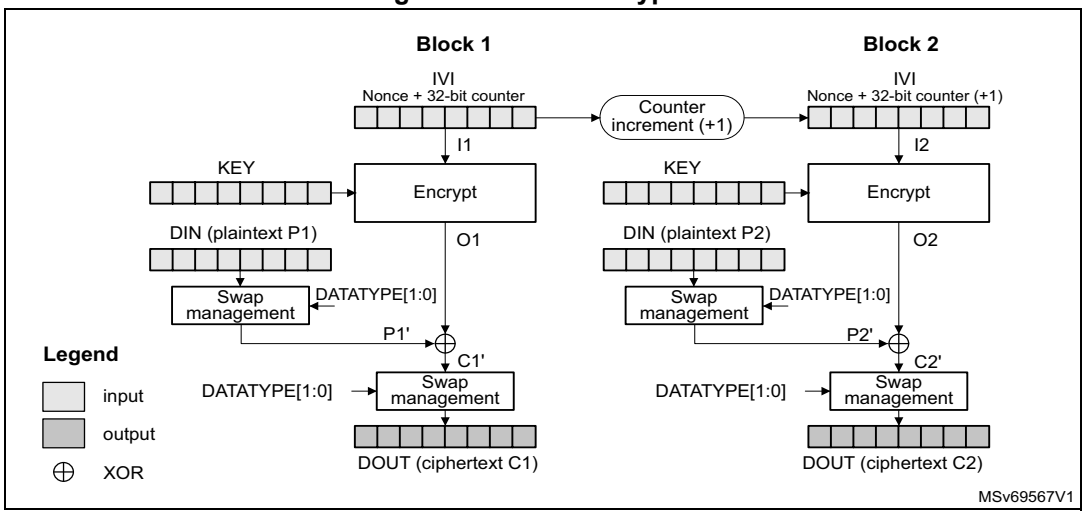
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 323 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 323. CTR encryption



Initialization vectors in SAES must be initialized as shown in *Table 333*.

Table 333. Counter mode initialization vector definition

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 34.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 34.4.14* (wrapped keys) and *Section 34.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to *Table 333*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 34.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.

8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in *Section 34.4.5*, then read the SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 34.4.9: SAES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

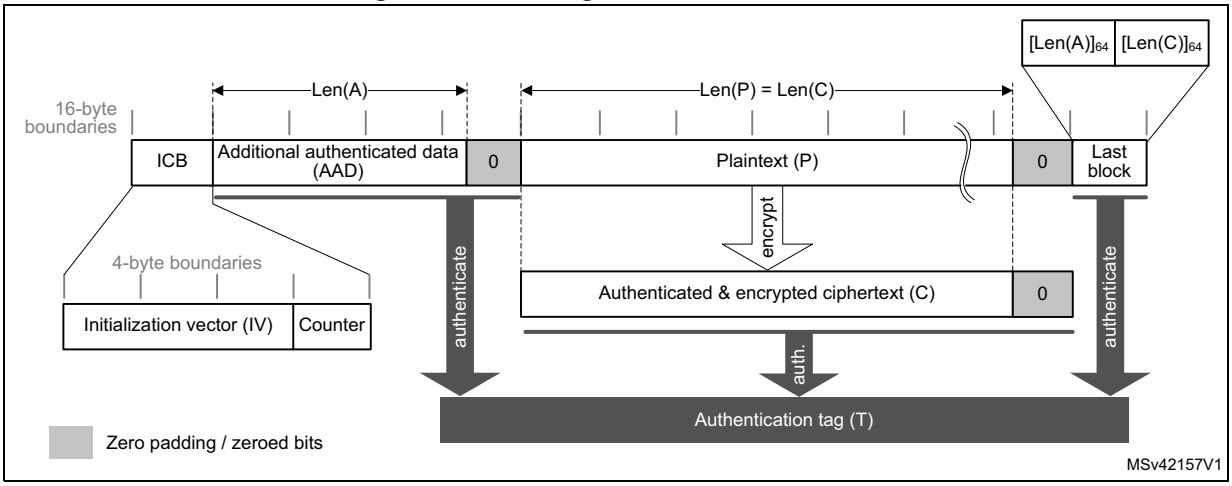
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

34.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 324. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed 2^{32} - 2 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 335*.

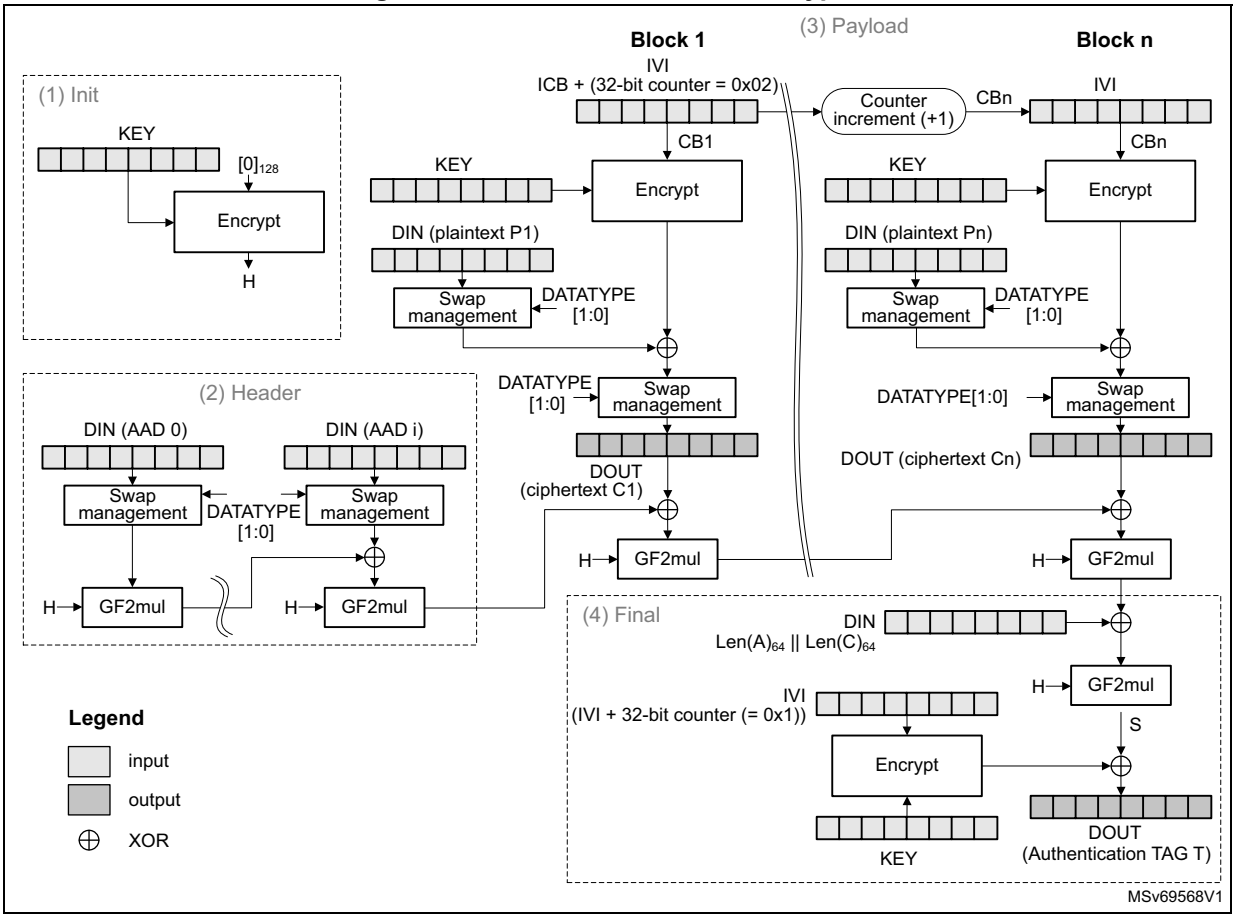
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 325 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 325. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 334*.

Table 334. Initialization of IV registers in GCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 335*.

Table 335. GCM last block definition

Word order to SAES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 34.4.6*, with the following sequence of events:

GCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (such as secure, nonsecure, specific CPU), also set KEYPROT.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 34.4.14* (wrapped keys) and *Section 34.4.15* (shared keys).
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to *Table 334*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 34.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

GCM header phase

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 34.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 34.4.5*, then read the SAES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.
15. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 335*.
17. Wait until the CCF flag in the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

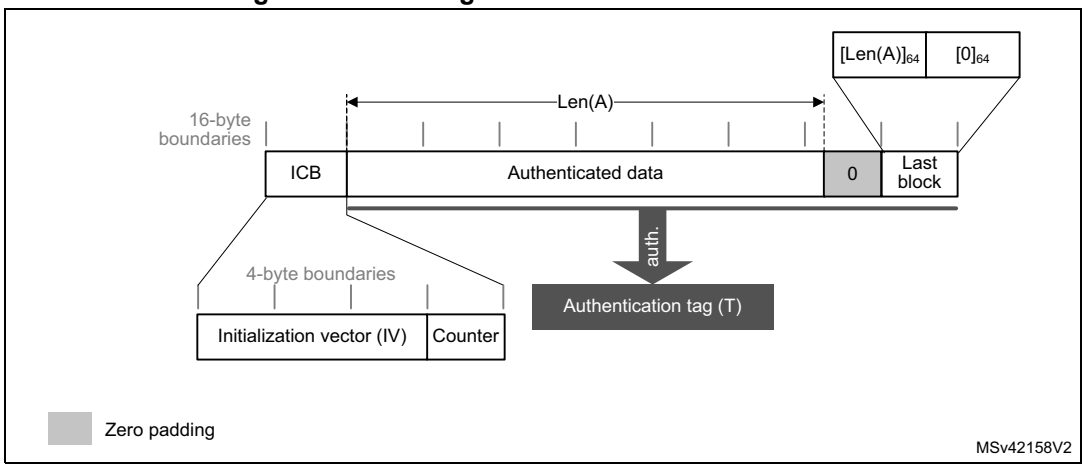
Suspend/resume operations are not supported in GCM mode.

34.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

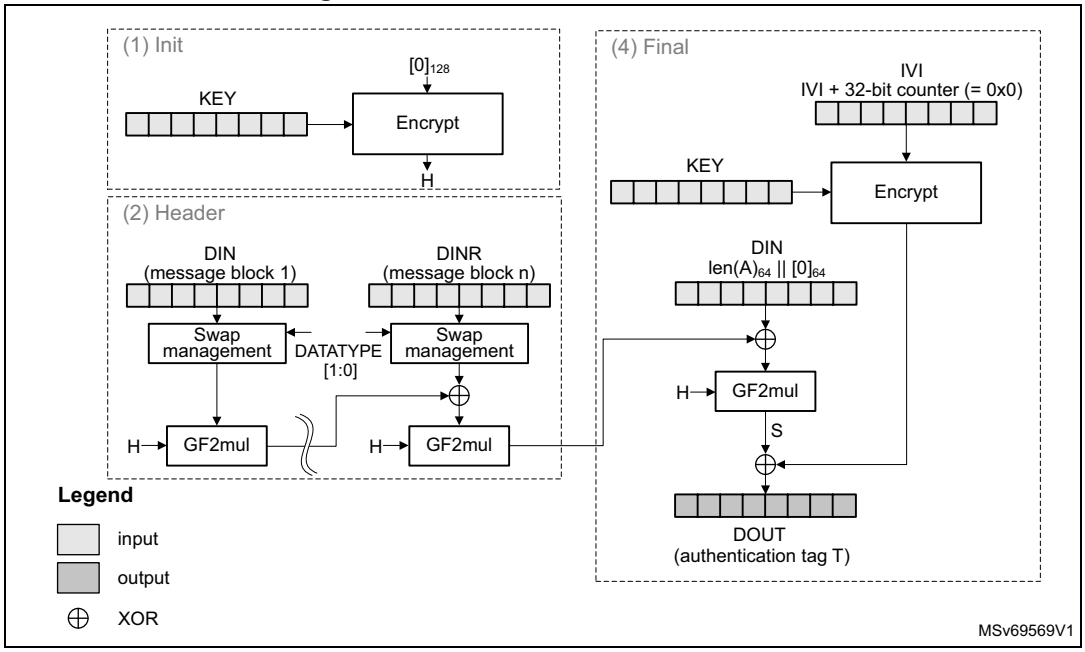
Figure 326. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 327 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 327. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

Suspend/resume operations are not supported in GMAC mode.

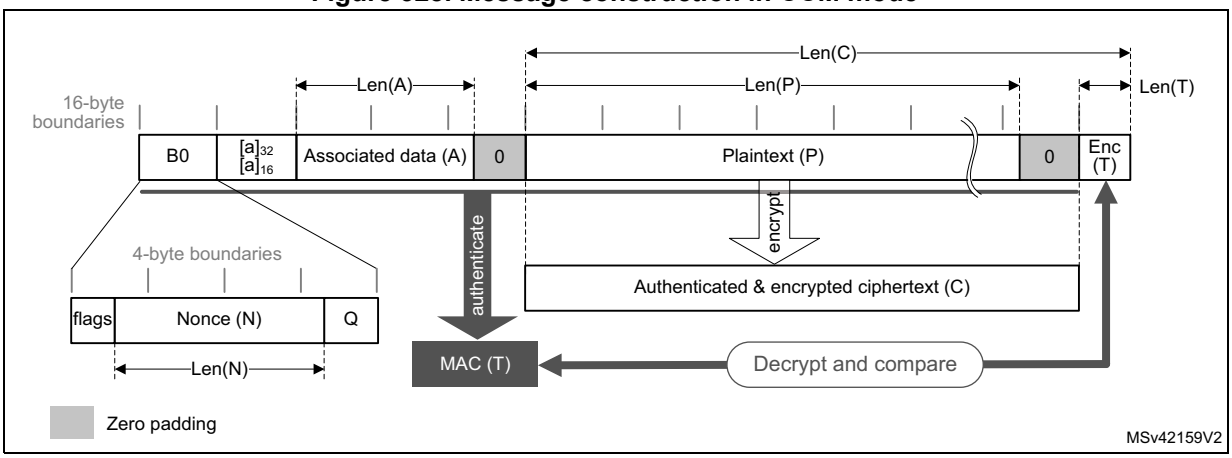
34.4.13 SAES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 328. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 328*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 328*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

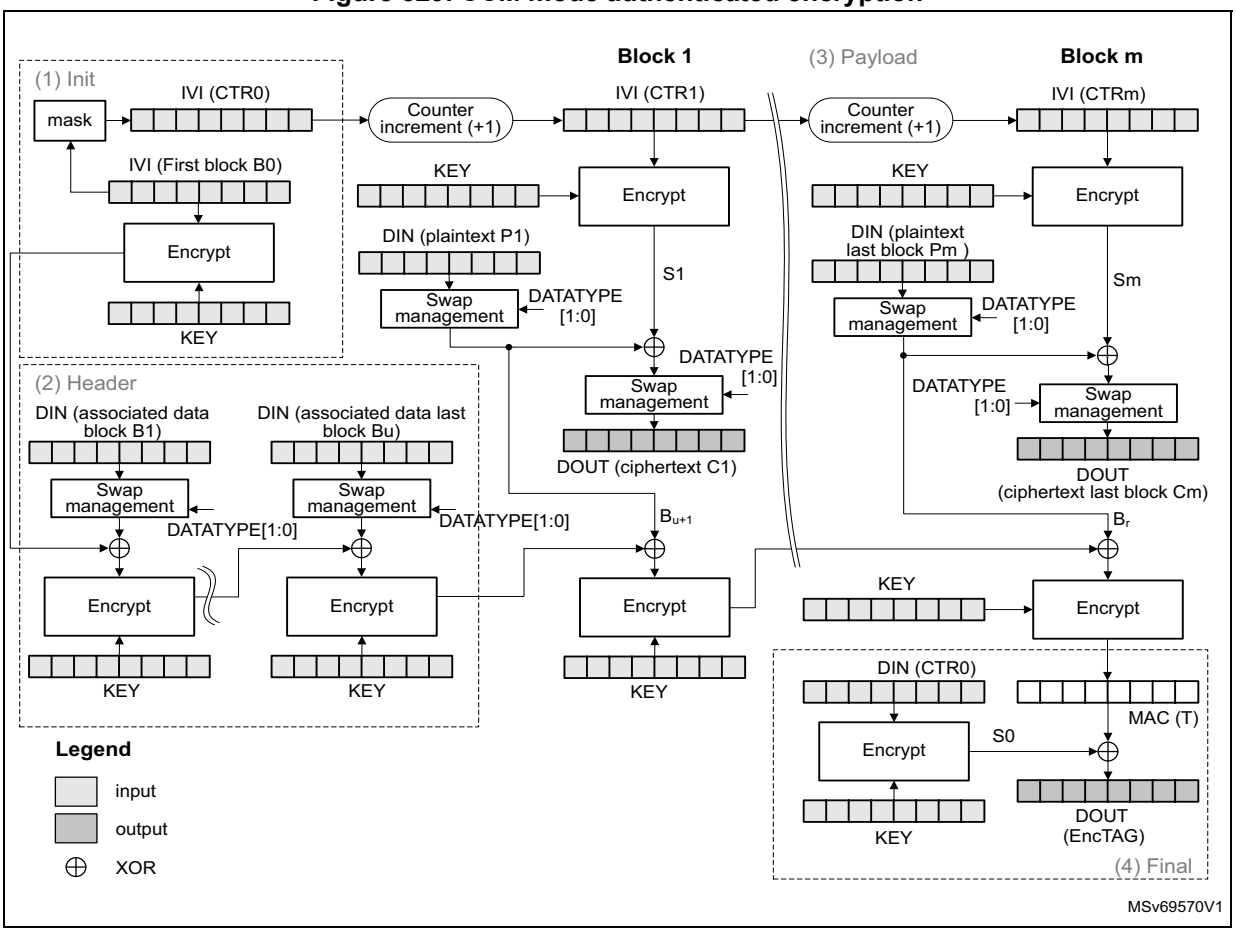
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C*, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 329 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 329. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 336*.

Table 336. Initialization of IV registers in CCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 34.4.6*, with the following sequence of events:

CCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context, also set the KEYPROT bit.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 34.4.14* (wrapped keys) and *Section 34.4.15* (shared keys).
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to *Table 336*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 34.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

CCM header phase

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 34.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral, by setting EN.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 34.4.5*, then read the SAES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
15. Wait until CCF flag in the SAES_ISR register is set.
16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation completed).
2. In the payload phase, read four times the SAES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.
4. Save the SAES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
3. Restore SAES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from the key registers, refer to *Section 34.4.17: SAES key registers* for details.
5. Enable the SAES peripheral, by setting EN.

34.4.14 SAES operation with wrapped keys

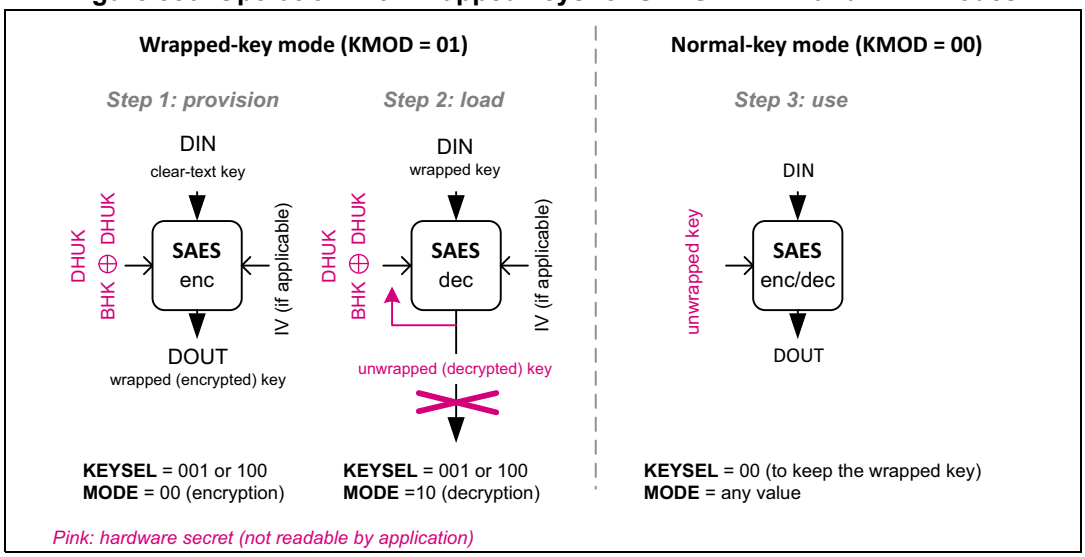
SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

Figure 330 summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 330. Operation with wrapped keys for SAES in ECB and CBC modes



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

Key wrapping for SAES

The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 34.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see *Table 338*).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register.
11. Repeat steps *8.* to *10.* if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.

Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in *key derivation* mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. With KEYSEL[2:0], select the same key source as when the key was wrapped/encrypted.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the *decryption* mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see *Table 338*).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
14. Repeat steps *12.* and *13.* if KEYSIZE is set.
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Decrypted wrapped key can be shared with an application running in a different security context (different security attribute) if KEYPROT bit was cleared during step 3.

Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERRF).

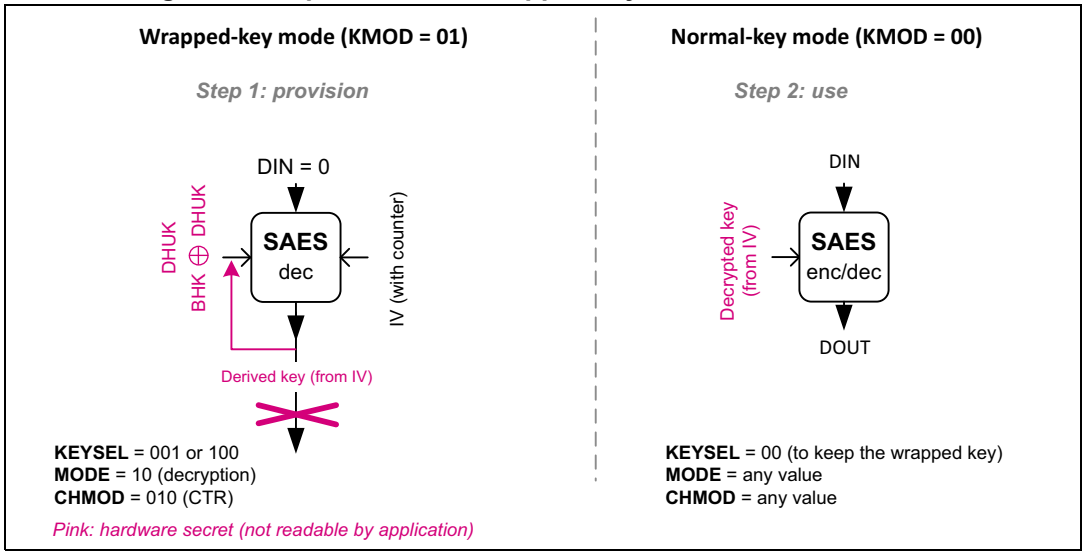
When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

Operation with wrapped keys for SAES in CTR mode

Figure 331 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by

writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 331. Operation with wrapped keys for SAES in CTR mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

The recommended sequence for SAES wrapped key mode using CTR is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 34.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps *8.* and *9.* if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Note: The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_*CR register.*

34.4.15 SAES operation with shared keys

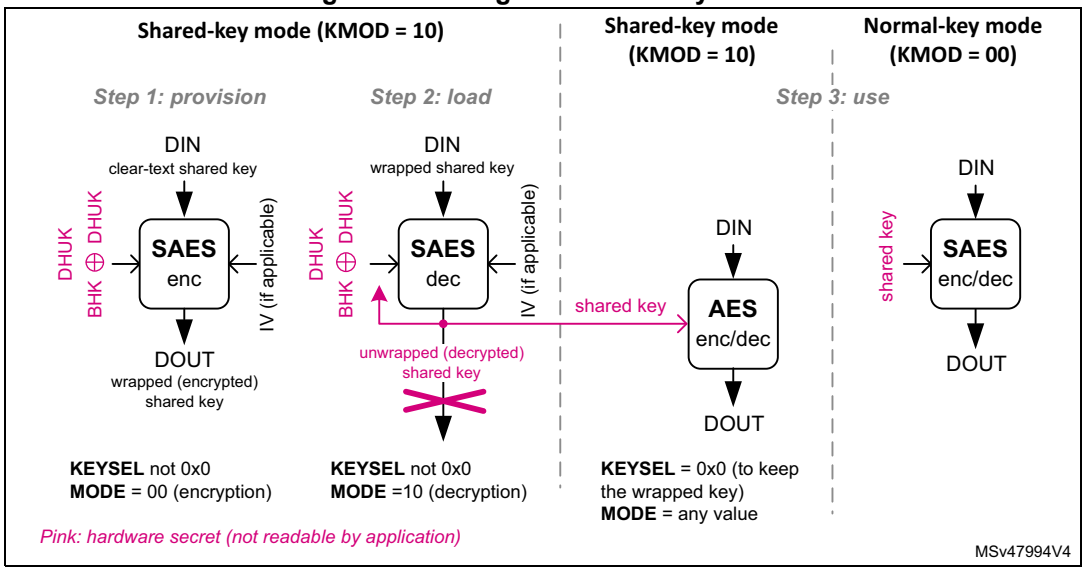
SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, the application software can make the AES keys available to the AES peripheral, without exposing them in clear-text (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

Note: When a key stored in SAES is shared with AES, the protection given by KEYPROT bit is lost. The protection is detailed in Section 34.4.17: SAES key registers.

Figure 332 summarizes how to wrap or unwrap keys to share with *AES* peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 332. Usage of Shared-key mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KSHAREID, KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

In the step 3, AES represents the AES peripheral.

Key wrapping for AES peripheral

Before SAES can share a key with the AES peripheral, the key must be encrypted (wrapped) once. The encryption sequence of a shared key is the same as for a wrapped

key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 332*. See *Key wrapping for SAES* for details.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected.

Key unwrapping for AES peripheral (shared key)

Each time SAES needs to share a key with the AES peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by AES. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 332*. See *Key unwrapping for SAES* for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERRF).

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be written with 0x0 (normal key mode).

Sequence in the AES peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the AES peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] at 0x2 and KEYVALID set. The sequence in the AES key share target peripheral is described in *AES key sharing with secure AES co-processor* of the corresponding section in this document. It can be run multiple times (for example, to manage a suspend/resume situation) as long as SAES is unused and duly remains in key sharing state.

Note: When KMOD[1:0] is at 0x2 and BUSY set in the AES peripheral, and KEYSIZE value of AES and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

34.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 337* “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output.

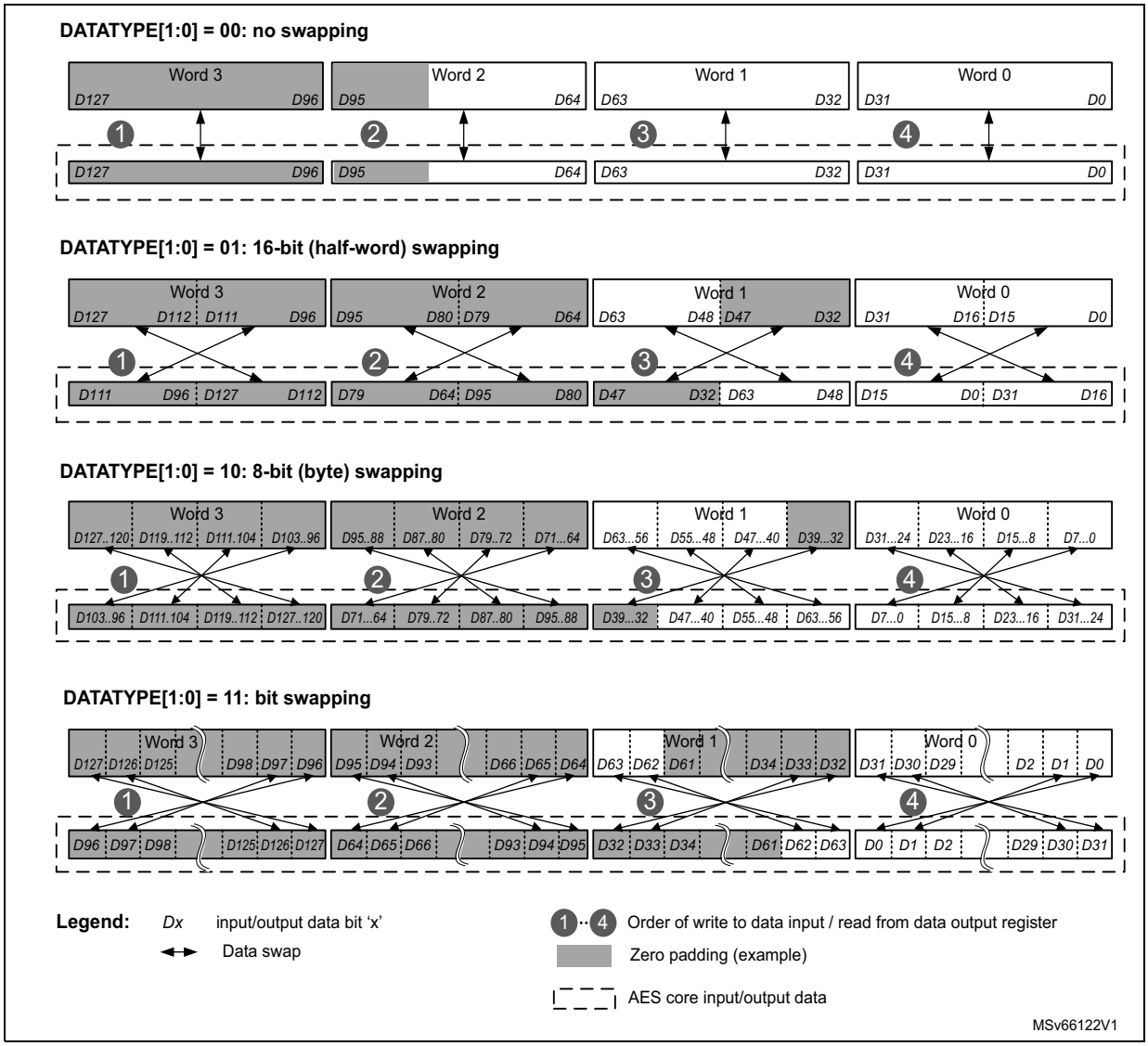
Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in *Table 337* and *Figure 333*.

Table 337. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 333. 128-bit block construction according to the data type



Data padding

Figure 333 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

34.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 338*. Reads are not allowed for security reason.

Note: In memory and in SAES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 338. Key endianness in SAES_KEYRx registers (128/256-bit keys)

SAES_KEYR7 [31:0]	SAES_KEYR6 [31:0]	SAES_KEYR5 [31:0]	SAES_KEYR4 [31:0]	SAES_KEYR3 [31:0]	SAES_KEYR2 [31:0]	SAES_KEYR1 [31:0]	SAES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
TAMP_BKP7R [31:0]	TAMP_BKP6R [31:0]	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 34.4.19*).

Key selection

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a non-volatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information. Thanks to the key derivation function, the computed DHUK depends on the SAES usage (privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], CPU, and KEYSIZE context).

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 338*.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR

registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 338*.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID is cleared. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transfered to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see *Section 34.4.19*).

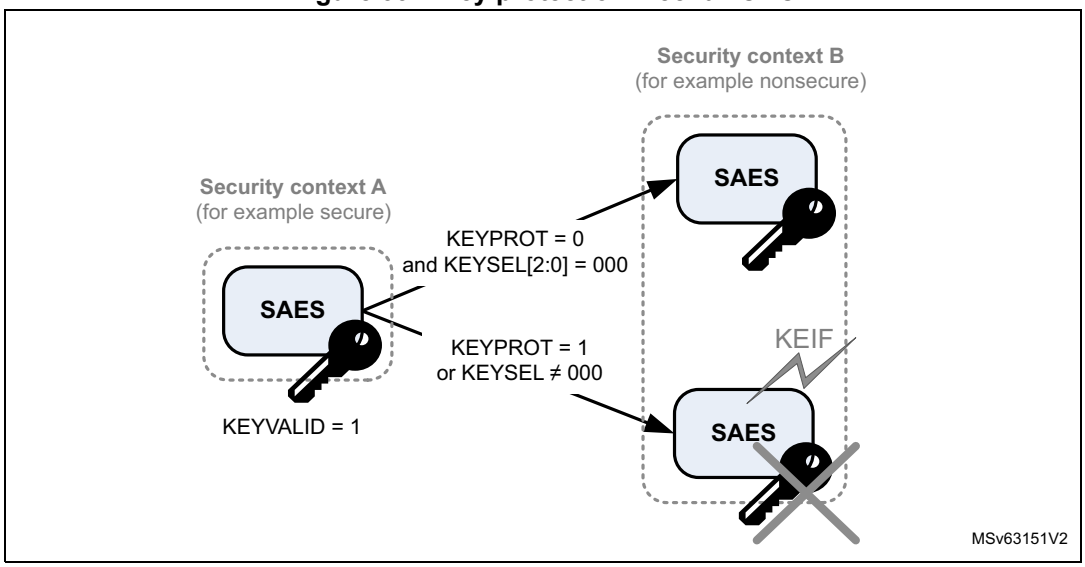
Note: DHUK, BHK and their XOR combination are not readable by any software (even secure).

Key protection

As depicted in *Figure 334*, when an application sets the KEYPROT bit before writing a key in SAES_KEYRx, any application executing in a different security context (that is, different security attribute) triggers a KEIF error flag upon access to any SAES register when KEYVALID is set.

Note: KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

Figure 334. Key protection mechanisms



34.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in *Table 339*. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 339. IVI bitfield spread over SAES_IVRx registers

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

34.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 34.5: SAES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 34.5: SAES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key protection error:** while KEYVALID is set, then if KEYPROT is set or KEYSEL[2:0] is different from zero, this error is triggered when an application executing in a security context different from the one used to load the key (that is, different security attribute) accesses SAES.

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 34.4.17: SAES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in AES peripheral does not match KEYSIZE in SAES peripheral.
- **Key sharing error:** triggered upon failing transfer of SAES shared key to AES peripheral. See *Section 34.4.15: SAES operation with shared keys* for details.
- **Hardware secret key loading error:** triggered upon failing load of DHUK or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK) or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 34.5: SAES interrupts*.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Upon a key sharing error, reset both AES and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring SAES.

RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the RNGEIE bit of the SAES_IER register is set. For more details, refer to *Section 34.5: SAES interrupts*.

Upon an RNG error:

- Make sure that the RNG peripheral is properly configured.
- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

Note: To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

Managing tamper errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, reset the SAES peripheral through RCC or global reset.

34.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See *Section 34.4.19: SAES error management* for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 340 gives a summary of the available features.

Table 340. SAES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

34.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with SAES can be found in *Appending data using DMA* subsection of *Section 34.4.5: SAES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 34.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

34.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 341. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
128-bit	Encryption or decryption(2)	ECB, CBC, CTR	**480**
	Key preparation	-	**145**
256-bit	Encryption or decryption(2)	ECB, CBC, CTR	**680**
	Key preparation	-	**230**

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

Table 342. Processing latency for GCM and CCM (in SAES kernel clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	490	72(2)	480(3)	490
		CCM	490	490	800	490
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	650	72(2)	690(3)	650
		CCM	650	680	1350	650

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.

34.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

34.8.1 SAES control register (SAES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:28 **KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x4: XOR of DHUK and BHK

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID is cleared.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared.

With the bitfield value other than zero and KEYVALID set, the application cannot transfer the ownership of SAES with a loaded key to an application running in another security context (such as secure, nonsecure). More specifically, when security of an access to any register does not match the information recorded by SAES, the KEIF flag is set.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 27:26 **KSHAREID[1:0]**: Key share identification

This bitfield defines, at the end of a decryption process with KMOD[1:0] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.

0x0: AES peripheral

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the SAES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.

0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

Others: Reserved

With KMOD[1:0] other than zero, any attempt to configure the SAES peripheral for use by an application belonging to a different security context (such as secure or nonsecure) results in automatic key erasure and setting of the KEIF flag.

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.

0: When KEYVALID is set and KEYSEL[2:0] = 0 application can transfer the ownership of the SAES, with its loaded key, to an application running in another security context (such as nonsecure, secure).

1: When KEYVALID is set, key error flag (KEIF) is set when an access to any registers is detected, this access having a security context (for example, secure, nonsecure) that does not match the one of the application that loaded the key.

Attempts to write the bit are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the key length in bits of the key used by SAES.

0: 128-bit

1: 256-bit

When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt.

Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode

This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the SAES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 34.4.16: SAES data registers and data swapping*.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data)
0x2: Byte swapping (8-bit data)
0x3: Bit-level swapping
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the SAES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the SAES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

34.8.2 SAES status register (SAES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

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

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

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 34.4.17: SAES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether SAES is idle or busy.

0: Idle

1: Busy

SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral.

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected write to SAES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 0 Reserved, must be kept at reset value.

34.8.3 SAES data input register (SAES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.

Reads return zero.

34.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set triggers a read error.

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

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

34.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.

Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

34.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.9 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in SAES_CR register

34.8.10 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

34.8.11 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

34.8.12 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

34.8.13 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.14 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.15 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.16 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

34.8.17 SAES suspend registers (SAES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

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

Bits 31:0 **SUSP[31:0]**: Suspend data

SAES_SUSPRx registers contain the complete internal register states of the SAES when the CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 34.4.8: SAES suspend and resume operations* for more details.

Clearing EN bit of the SAES_CR register clears this register to zero.

SAES_SUSPRx registers are not used in other chaining modes than CCM.

34.8.18 SAES interrupt enable register (SAES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

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

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

Bit 3 **RNGEIE:** RNG error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

34.8.19 SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												r	r	r	r

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

Bit 3 **RNGEIF:** RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected

1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in the SAES_SR register.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:

–SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4).

–SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4).

–AES fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2).

–KEYVALID is set and either KEYPROT is set or KEYSEL[2:0] is other than 0x0. The security context of the application that loads the key (secure or nonsecure) does not match the security attribute of the access to SAES_CR or SAES_DOUT. In this case, KEYVALID and EN bits are cleared.

–SAES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the SAES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

34.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												w	w	w	w

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

Bit 3 **RNGEIF:** RNG error interrupt flag clear

Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.

Bit 2 **KEIF:** Key error interrupt flag clear

Setting this bit clears the KEIF status bit of the SAES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear

Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 0 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit of the SAES_ISR register.

34.8.21 SAES register map

Table 343. SAES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SAES_CR	IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1]	CHMOD[0]	MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	SAES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	SAES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SAES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SAES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	SAES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SAES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SAES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SAES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SAES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SAES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SAES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SAES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SAES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SAES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SAES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 343. SAES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SAES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SAES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SAES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SAES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SAES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	SAES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	SAES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	SAES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	SAES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
	Reset value																													0	0	0	0
0x304	SAES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x308	SAES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x309-0x3FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

35 Hash processor (HASH)

35.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512).

35.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO, corresponding to a 64-byte block size (16 x 32 bits)
- Automatic padding to complete the input bit string to fit digest minimum block size
- AHB slave peripheral, accessible by 32-bit words only (else an AHB error is generated)
- 8 × 32-bit words (H0 to H15) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers
  - Hashing computation suspend/resume mechanism, including DMA

35.3 HASH implementation

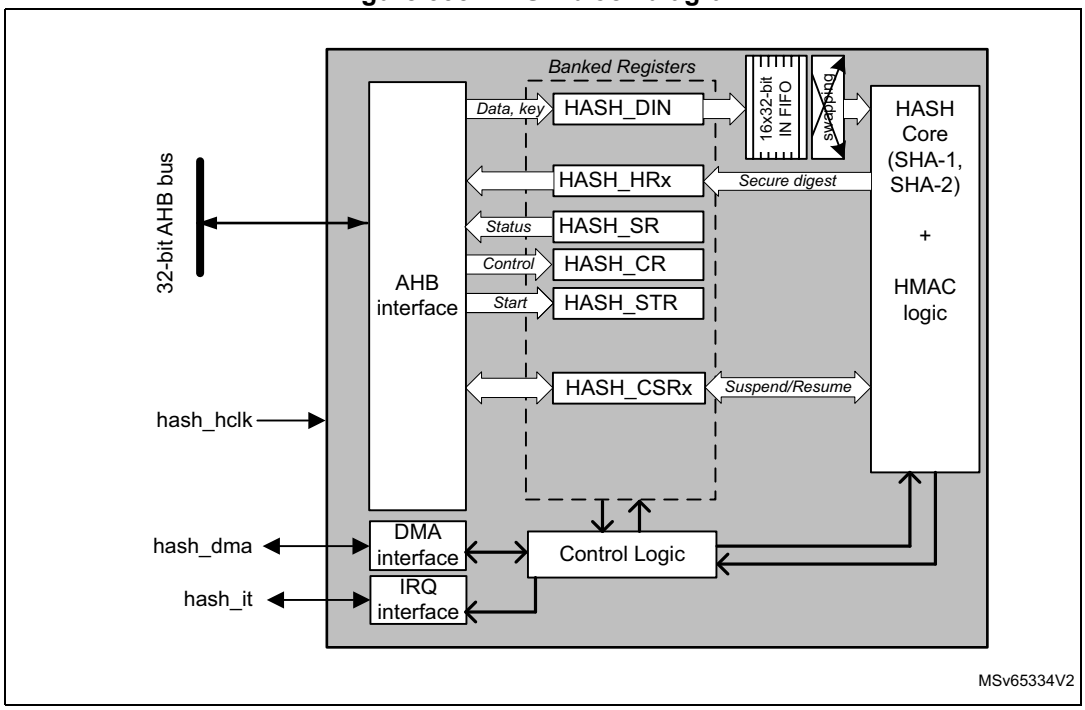
The devices have a single instance of HASH peripheral.

35.4 HASH functional description

35.4.1 HASH block diagram

Figure 335 shows the block diagram of the hash processor.

Figure 335. HASH block diagram



35.4.2 HASH internal signals

Table 344 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 344. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

35.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 345*).

Table 345. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)(1)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA2-224	224			
SHA2-256	256			
SHA2-384	384	128	$< 2^{128}$ bits	Yes
SHA2-512	512(2)			

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

35.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

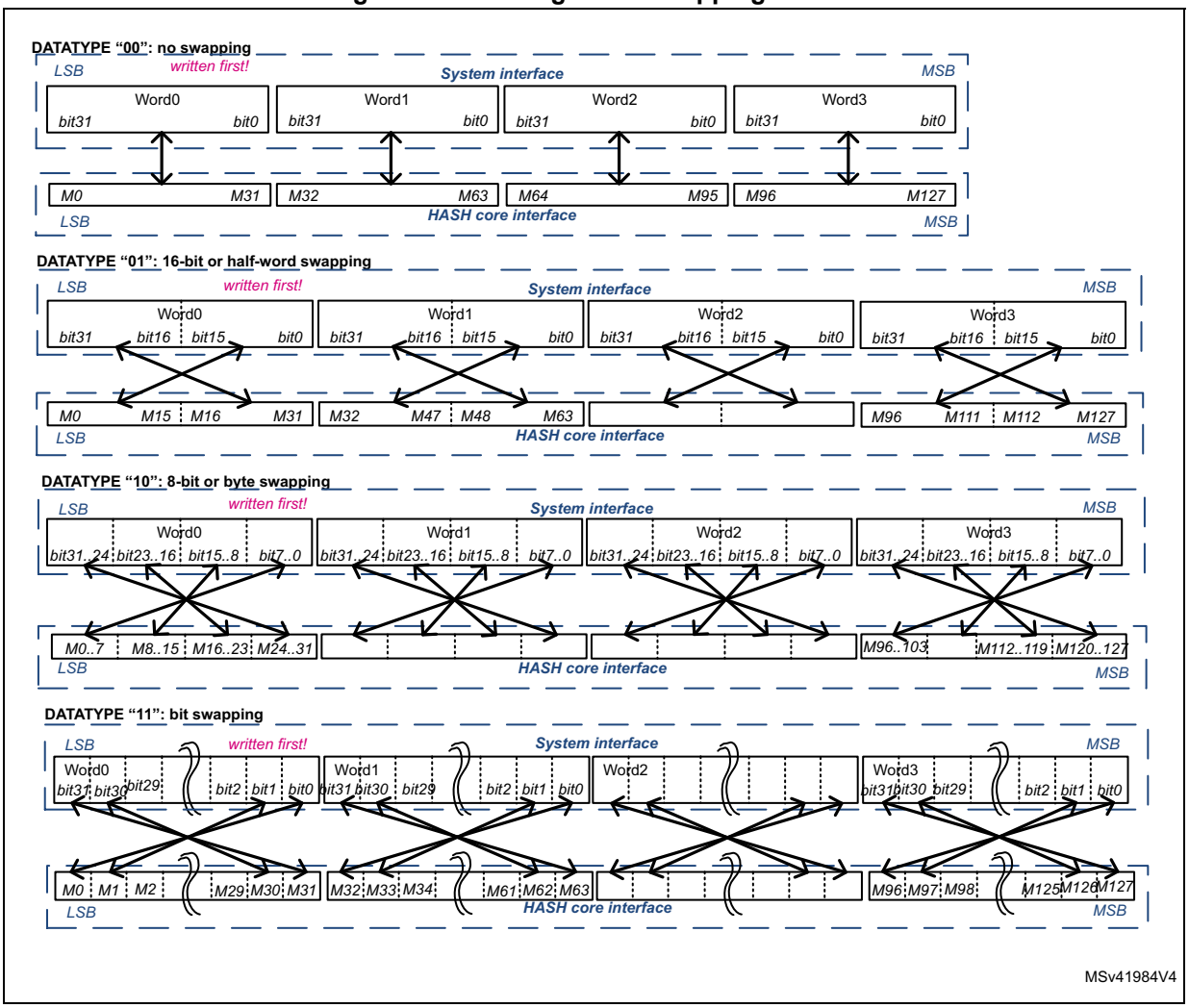
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 336* shows how the hash processing core 32-bit data block M0...31 is

constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 336. Message data swapping feature



35.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 345: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 35.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the

input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals **`(NBWE-1) * 4`**. This step is not required if the block size is already known (see *Table 345* for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 35.4.7: HMAC operation* for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 35.4.6* for details).
   - Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 35.4.6* for details).
   - Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to

receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 346*. When set, the DMAEN bit is automatically cleared by hardware.

Table 346. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256
SHA2-384	HASH_H0 to HASH_H11	HASH_H0[31]	384
SHA2-512	HASH_H0 to HASH_H15		512(1)

1. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to *Section 35.4.7: HMAC operation*.

35.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed hashing sequences with padding are described in *Section 35.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a "**1**" is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

01100001 01100010 01100011 *1*UUUUUUU

Since L = 24, the number of bits in the above bit string is 25, and 423 "0" bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

6162638*0* 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where '**U**' means don't care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D

35.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

HMAC(message)= Hash((Key | pad) XOR opad |
Hash((Key | pad) XOR ipad | message))

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.
3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 35.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 35.4.11: HASH processing time.

Endianness management details can be found in Section 35.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   ```
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F
   ```

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

   ```
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU
   ```

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

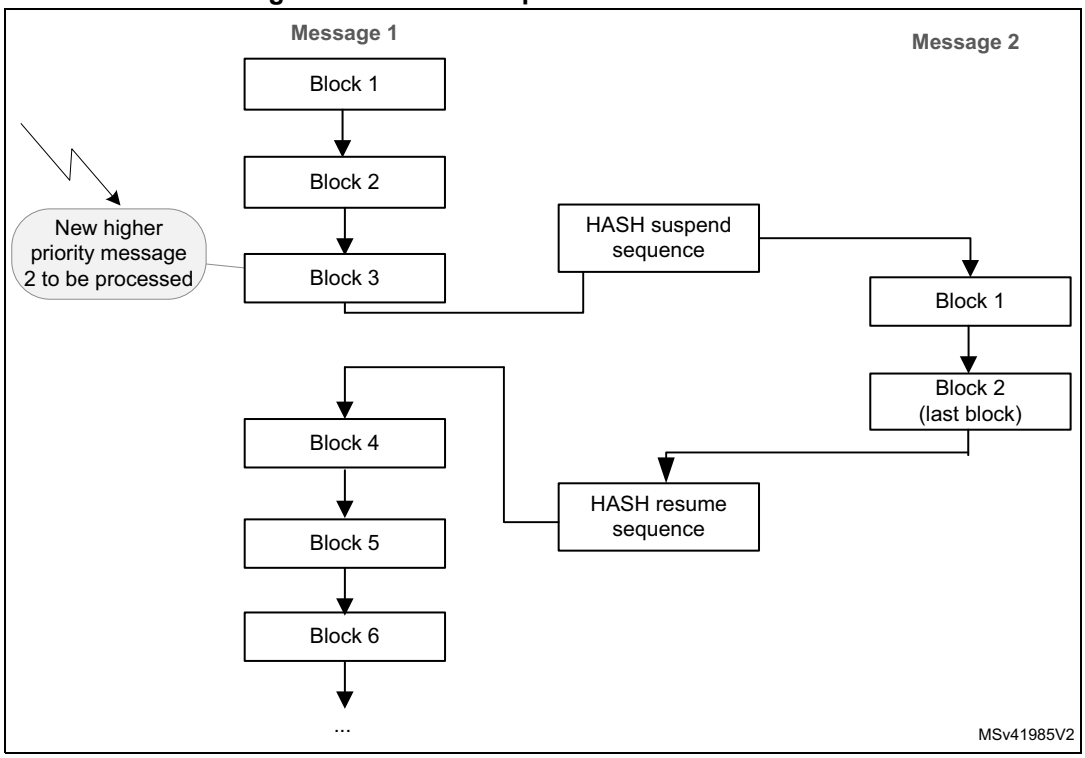
   ```
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
   ```

35.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 337*.

Figure 337. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 337*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

35.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 35.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

35.4.10 HASH error management

No error flags are generated by the hash processor.

35.4.11 HASH processing time

Table 347 summarizes the time required to process an intermediate block for each mode of operation.

Table 347. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA2-224		16	50	**66**
SHA2-256				
SHA2-384	128	32	66	**98**
SHA2-512(2)				

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

35.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 348* gives a summary of the available features.

Table 348. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

35.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

35.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:0]				LKEY
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	rw		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

0000: SHA-1

0001: reserved

0010: SHA2-224

0011: SHA2-256

1100: SHA2-384

1101: SHA2-512/224

1110: SHA2-512/256

1111: SHA2-512

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see *Table 345: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bit is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

Setting this bit resets the hash processor core, so that the HASH is ready to compute the message digest of a new message.
Clearing this bit has no effect. Reading this bit always returns 0.

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

35.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see *Table 345: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

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

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.

Do not write to this register when BUSY is set.

Reading this register returns zeros.

35.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Reading this bit returns zero.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

35.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

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

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 35.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 35.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x324 + 0x4 * x, (x = 5 to 15)

Reset value: 0x0000 0000

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

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 35.6.4: HASH digest registers* introduction.

35.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

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

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

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

35.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

35.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 102)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0020 0000 (HASH_CSR2)

Reset value: 0x0000 0000 (others)

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

Bits 31:0 **CSx[31:0]:** Context swap x

Refer to *Section 35.6.7: HASH context swap registers* introduction.

35.6.8 HASH register map

Table 349. HASH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:2]		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NBWE[4:0]					DINNE		NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												1	0	0	0	1	0		0	0	0	0	0						0	0	0	1
Reserved																																	

Table 349. HASH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x290	HASH_CSR102	CSR102[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 4) Last address: 0x320	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x324 + 0x4 * x, (x = 5 to 15) Last address: 0x34C	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

36 Public key accelerator (PKA)

36.1 PKA introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, all needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

When manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

36.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)

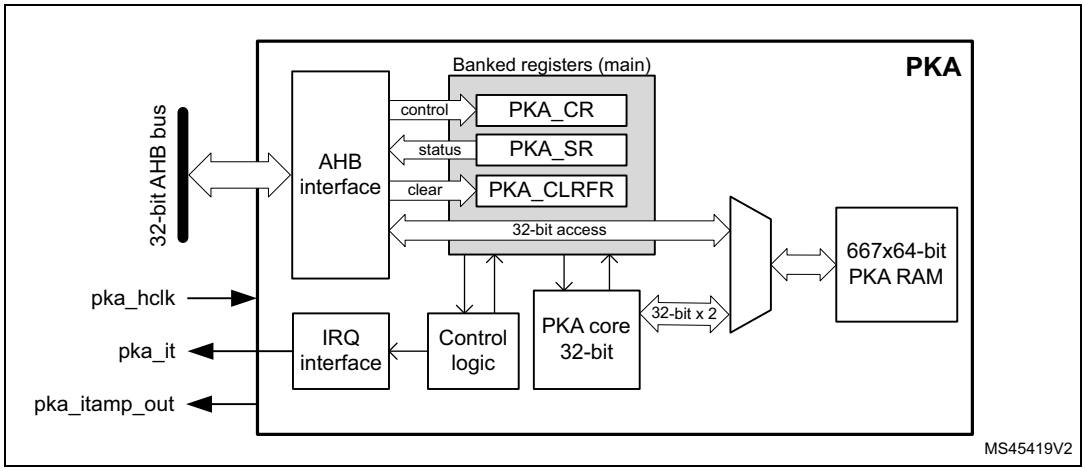
36.3 PKA implementation

The devices have a single instance of the peripheral.
The STM32H523/562/563xx devices use the PKA in limited mode (only ECDSA signature verification is available).

36.4 PKA functional description

36.4.1 PKA block diagram

Figure 338. PKA block diagram



36.4.2 PKA internal signals

Table 350 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 350. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

36.4.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 36.4.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

36.4.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 36.6.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 351* and *Table 352*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 351. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R2 mod n	*Section 36.5.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 36.5.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 36.5.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 36.5.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 36.5.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 36.5.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 36.5.7*
0x0D	001101	Modular reduction A mod n	*Section 36.5.8*
0x09	001001	Arithmetic addition A+B	*Section 36.5.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 36.5.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 36.5.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 36.5.12*
0x07	000111	RSA CRT exponentiation	*Section 36.5.13*

Table 352. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 36.5.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 36.5.15*
0x23	100011	ECC complete addition	*Section 36.5.18*
0x24	100100	ECDSA sign (protected)	*Section 36.5.16*
0x26	100110	ECDSA verification	*Section 36.5.17*
0x27	100111	ECC double base ladder	*Section 36.5.19*
0x2F	101111	ECC projective to affine	*Section 36.5.20*

Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register. If the application selects any value that is not documented below the write to MODE bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in *Table 351* and *Table 352* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is supplied before starting the operation. Performance improvement is detailed in *Section 36.6.2: Computation times*.

The only operation using fast mode is modular exponentiation (MODE = 0x02).

36.4.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

Specifications of the above functions are given in following publications:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

RSA key pair

For the following RSA operations a public key and a private key information are defined as below:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information on the above representations refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*) must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) * (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
4. Convert the integer message representative *m* to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

Above operations can be accelerated by PKA using *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process protected version of modular exponentiation (MODE = 0x3) is strongly recommended for security reason. For encryption process MODE = 0x3 cannot be used, as it requires the knowledge of the private key.

Elliptic curve selection

For following ECC operations curve parameters are defined as below:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 36.6.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, *n* x G = identity element O).

ECDSA message signature generation

ECDSA (elliptic curve digital signature algorithm) signature generation function principle is the following: Alice, to sign a message *m* using her private key integer d_A, goes through the following steps.

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

Steps 4 to 7 are accelerated by PKA using:

- *ECDSA sign* or
- All of the operations below:
  - *ECC Fp scalar multiplication* k x P
  - *Modular reduction* A mod *n*
  - *Modular inversion* A^{-1} mod *n*
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A.

Bob can verify that Q_A is a valid curve point going through the following steps:

1. check that Q_A is not equal to the identity element O
2. check that Q_A is on the agreed curve
3. check that n x Q_A = O.

Then Bob follows the procedure detailed below:

1. verify that *r* and *s* are integer in [1, n-1]
2. calculate *e* = HASH(*m*), where HASH is the agreed cryptographic hash function
3. let z be the L_n leftmost bits of *e*
4. calculate w = s^{-1} mod *n*
5. calculate u_1 = zw mod *n* and u_2 = *rw* mod *n*
6. calculate the curve point (x_1, y_1) = u_1 x G + u_2 x Q_A
7. the signature is valid if *r* = x_1 (mod *n*), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using *ECDSA verification*.

36.4.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. The PKA becomes available when INITOK bit is set in PKA_SR. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

Note: When PKA is in the process of clearing its memory EN bit cannot be set.

When setting EN bit in PKA_CR make sure that the value of MODE bitfield corresponds to an authorized PKA operation (see OPERRF in Section 36.4.7).

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only PKA registers can be accessed, with writes to EN bits ignored.

If INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 36.5*.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error, and step *3* (PROCENDF = 1) never happens. See *Section 36.4.7* for details.

Using precomputed Montgomery parameters (PKA Fast mode)

As explained in *Section 36.4.4*, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, *Section 36.5.5*). This is know as “Fast mode”.

To manage the usage of Fast mode it is recommended to follow the procedure described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in *Section 36.6.1*.
2. Program in PKA_CR register the PKA in *Montgomery parameter computation* mode (MODE=”0x1”) then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in *Section 36.5*.

36.4.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the address error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM error flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return 0, while writes are ignored.
- The selected operating mode using MODE bitfield is not listed in PKA operating modes (see bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). In this case the operation error flag (OPERRF) is set in the PKA_SR register, and write to MODE bitfield is ignored.

For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see *Section 36.7* for details).

ADDRERRF, OPERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

OPERRF error must be cleared using OPERRFC bit in PKA_CLRFR before a new operation is written in PKA_CR register.

Note: The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

36.5 PKA operating modes

36.5.1 Introduction

The various operations supported by PKA are described in the following subsections, defining the format of the input data and of the results, both stored in the PKA RAM.

Warning: The validity of all input parameters to the PKA must be checked before starting any operation, as PKA assumes that all of them are valid and consistent with each other. Input parameters must not exceed the operand size specified in the operation tables.

The following information applies to all PKA operations.

- PKA core processes 64-bit words in its RAM. Hence hereafter all word size is 64-bit
- When an element is written as input in the PKA RAM, an additional word with all bits equal to zero has to be added after the most significant input word. This rule does not apply if the operand has a fixed size of 1.
- All reported RAM storage addresses refer to the least significant word of the data, and to obtain the actual address to use application must add to the indicated offset the base address of the PKA.
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is (*rsa_size* / 64 + 1) words, with *rsa_size* equal to the chosen modulus length in bits. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 17 words, or 1088 bits.
  - EOS (ECC operand size): data size is (*ecc_size* / 64 + 1) words, with *ecc_size* equal to the chosen prime modulus length in bits. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 4 words, or 256 bits.
  - ROS and EOS values include the required additional all 0 word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Fractional results for above formulas must be rounded up to the nearest integer since PKA core processes 64-bit words.

Note: The maximum ROS is 66 words (4160-bit max exponent size), while the maximum EOS is 11 words (640-bit max operand size).

As a first example (and to better understand the endianess in PKA memory), to prepare the operation *ECC Fp scalar multiplication*, when the application writes the x coordinate of point

P for an ECC P256 curve (EOS = 5 words), the least significant bit must be placed in bit 0 at address offset 0x578, and the most significant bit in bit 63 at address offset 0x590. Then, as mentioned above, the application must write the empty word 0x0000000000000000 at address offset 0x598.

As a second example, still to prepare the operation ECC Fp scalar multiplication, when the application need to write the information a = -3, on a curve with a modulus length of 224 bits (that is, four 64-bit words, rounded up, plus one) following data must be written in PKA memory:

```
@RAM+410  0x0000000000000001  /* curve coefficient 'a' sign without extra word */
@RAM+418  0x0000000000000011  /* value of |a| LSB
@RAM+420  0x0000000000000000  ...
@RAM+428  0x0000000000000000  ...
@RAM+430  0x0000000000000000  value of |a| MSB */
@RAM+438  0x0000000000000000  /* additional all 0 word */
```

36.5.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod *n*) used by PKA to convert operands into the Montgomery residue system representation. This operation can be very useful when fast mode operation is used, because in this case the Montgomery parameter is passed as input, saving the time for its computation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in *Table 353*.

Table 353. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0x1088	ROS
OUT	Result: R^2 mod n	-	RAM@0x620	

36.5.3 Modular addition

Modular addition operation consists in the computation of A + B mod *n*. Operation instructions are summarized in *Table 354*.

Table 354. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A+B mod n	(0 ≤ result < n)	RAM@0xE78	

36.5.4 Modular subtraction

Modular subtraction operation consists in the following computations:

- If A ≥ B result equals A - B mod *n*
- If A < B result equals A + n - B mod *n*

Operation instructions are summarized in *Table 355*.

Table 355. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A-B mod n	(0 ≤ result < n)	RAM@0xE78	

36.5.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication A x B mod *n*

The method to perform above operations are described below. Note that “x” function is this operation, and A, B, C operands are in the natural domain.

1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Assuming that A is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – Result AR= A x *r2modn* mod *n* is A in the Montgomery domain.
   b) Assuming that BR is an integer in the Montgomery domain:
      – Result B = BR x 1 mod *n* is B in the natural domain.
      – Similarly, above value AR computed in a) can be converted into the natural domain by computing A = AR x 1 mod *n*.
2. Simple modular multiplication A x B mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain.
   c) Compute AB= AR x B mod *n*. Output is in natural domain.
3. Multiple modular multiplication A x B x C mod *n*
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute AR = A x *r2modn* mod *n*. Output is in the Montgomery domain.
   c) Compute BR = B x *r2modn* mod *n*. Output is in the Montgomery domain.
   d) Compute ABR= AR x BR mod *n*. Output is in the Montgomery domain.
   e) Compute CR = C x *r2modn* mod *n*. Output is in the Montgomery domain.
   f) Compute ABCR= ABR x CR mod *n*. Output is in the Montgomery domain.
   g) (optional) Repeat the two steps above if more operands need to be multiplied.
   h) Compute ABC= ABCR x 1 mod *n* to retrieve the result in natural domain.

Operation instructions for Montgomery multiplication are summarized in *Table 356*.

Table 356. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Operand B	(0 ≤ B < n)	RAM@0xC68	
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: AxB mod n(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

36.5.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of A^e mod *n*.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 357* and *Table 358*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 359*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 357* (normal mode), *Table 358* (fast mode) and in *Table 359* (protected mode). Fast mode usage is explained in *Section 36.4.6*.

Table 357. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 358. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0xC68	ROS
IN	Exponent e	(0 ≤ e < n)	RAM@0xE78	
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
IN/OUT	Montgomery parameter R2 mod n	(mandatory)	RAM@0x620	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	

Table 359. Modular exponentiation (protected mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x03	PKA_CR	6 bits
	Exponent e length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand A (base of exponentiation)	(0 ≤ A < n)	RAM@0x16C8	ROS
	Exponent e	(0 ≤ e < n)	RAM@0x14B8	
	Modulus value n	(odd integer only, n < 2^{4096})	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod n	(0 ≤ result < n)	RAM@0x838	
ERROR	Error A^e mod n	No errors: 0xD60D Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n^1 with phi = (p - 1) * (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (*p*, *q*, *e*, *d*) instead of (*N*, *e*, *d*, *phi*), in this case, to derive *N* and *phi* using PKA arithmetic multiplier: *N* = *p* * *q*, and *phi* = (*p* - 1) * (*q* - 1).

36.5.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse A^{-1} mod *n*. If the modulus *n* is prime, for all values of A (1 ≤ A < *n*) modular inversion output is valid. If the modulus *n* is not prime, A has an inverse only if the greatest common divisor between A and *n* is 1.

If the operand A is a divisor of the modulus *n* the result is a multiple of a factor of *n*.

Operation instructions for modular inversion are summarized in *Table 360*.

Table 360. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand A	(0 ≤ A < n)	RAM@0xA50	ROS
	Modulus value n	(odd integer only, n < 2^{4160})	RAM@0xC68	
OUT	Result: A^{-1} mod n	0 < result < n	RAM@0xE78	

36.5.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of A divided by n. Operation instructions are summarized in *Table 361*.

Table 361. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand A	($0 \le A < 2n < 2^{4160}$)	RAM@0xA50	ROS
	Modulus value n	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result A mod n	(0 < result < n)	RAM@0xE78	

36.5.9 Arithmetic addition

Arithmetic addition operation consists in the computation of A + B. Operation instructions are summarized in *Table 362*.

Table 362. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x09	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: A+B	($0 \le$ result $< 2^{M+1}$)	RAM@0xE78	ROS + 1

36.5.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:

- If $A \ge B$ result equals A - B
- If A < B and M/32 residue is > 0 result equals $A + 2^{int(M/32)*32+1} - B$
- If A < B and M/32 residue is 0 result equals $A + 2^{int(M/32)*32} - B$

For the last two bullets the 32-bit word following the most significant word of the output equals 0xFFFF FFFF, as result is negative.

Operation instructions are summarized in *Table 363*.

Table 363. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0A	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le < A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le < B < 2^M$)	RAM@0xC68	
OUT	Result: A-B	($0 \le$ result $< 2^M$)	RAM@0xE78	

36.5.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in *Table 364*.

Table 364. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0B	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: AxB	($0 \le result < 2^{2M}$)	RAM@0xE78	2xROS

36.5.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:

- If A = B then result = 0xED2C
- If A > B then result = 0x7AF8
- If A < B then result = 0x916A

Operation instructions for arithmetic comparison are summarized in *Table 365*.

Table 365. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x0C	PKA_CR	6 bits
	Operand length M	(in bits, not null)	RAM@0x408	64 bits
	Operand A	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand B	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result A?B	0xED2C, 0x7AF8 or 0x916A	RAM@0xE78	64 bits

36.5.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation $m = A^d \pmod{pq}$ more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation $A^d \bmod pq$ are summarized in *Table 366*.

Table 366. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	$(0 < d_P < 2^{M/2})$	RAM@0x730	ROS / 2
	Operand d_Q	$(0 < d_Q < 2^{M/2})$	RAM@0xE78	
	Operand q_{inv}	$(0 < q_{inv} < 2^{M/2})$	RAM@0x948	
	Prime p(1)	$(0 < p < 2^{M/2})$	RAM@0xB60	
	Prime q(1)	$(0 < q < 2^{M/2})$	RAM@0x1088	
IN	Operand A	$(0 < A < 2^{M})$	RAM@0x12A0	ROS
OUT	Result: $A^d \bmod pq$	$(0 \le \text{result} < pq)$	RAM@0x838	

1. Must be different from 2.

36.5.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in *Table 367*.

Table 367. Point on elliptic curve Fp check

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(|b| < p)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < p < 2640)	RAM@0x470	
	Point P coordinate *x*	(x < *p*)	RAM@0x578	
	Point P coordinate *y*	(y < *p*)	RAM@0x5D0	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Result: point P on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: x or y coordinate is not smaller than modulus p	RAM@0x680	64 bits

36.5.15 ECC Fp scalar multiplication

This operation consists in the computation of k x P (x_P, y_P), where P is a point on a curve over prime fields and “x” is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in *Table 368*.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 368. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 368. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier k	($0 \leq k < 2^{640}$)	RAM@0x12A0	
	Point P coordinate x_P	($x < p$)	RAM@0x578	
	Point P coordinate y_P	($y < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: k x P coordinate *x'*	(result < *p*)	RAM@0x578	
	Result: k x P coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error k x P	No errors: 0xD60D Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For k = 0 this function returns a point at infinity (0, 0) if curve parameter b is nonzero, (0, 1) otherwise. For k different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation must be carried out.
- For k < 0 (that is, a negative scalar multiplication is required) the multiplier absolute value
  k = |-k| must be provided to the PKA. After the computation completion, the formula -P = (x, -y) can be used to compute the y coordinate of the effective final result (the x coordinate remains the same).

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

36.5.16 ECDSA sign

ECDSA signing operation (outlined in *Section 36.4.5*) is summarized in *Table 369* (input parameters) and in *Table 370* (output parameters).

The application has to check if the output error is equal to 0xD60D, if it is different a new k must be generated and the ECDSA sign operation must be repeated.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 369. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x24	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus p length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x418	EOS
	Curve coefficient b	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer k(1)	($0 \leq k < 2^{640}$)	RAM@0x12A0	
	Curve base point G coordinate x	(x < *p*)	RAM@0x578	
	Curve base point G coordinate y	(y < *p*)	RAM@0x470	
	Hash of message z	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key d	(0 < d)	RAM@0xF28	
	Curve prime order n	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases k can be deterministically generated.
2. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 370. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	(0 < *r* < n)	RAM@0x730	EOS
	Signature part *s*	(0 < *s* < n)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature (*Table 369* and *Table 370*, respectively), with the addition of the coordinates of the point kG. This extra output is defined in *Table 371*.

Table 371. Extended ECDSA sign - Extra outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point kG coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point kG coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

36.5.17 ECDSA verification

ECDSA verification operation (outlined in *Section 36.4.5*) is summarized in *Table 372* (input parameters) and *Table 373* (output parameters).

The application has to check if the output error is equal to 0xD60D, if different the signature is not verified.

Table 372. ECDSA verification - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x26	PKA_CR	6 bits
	Curve prime order n length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus p length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient a sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient |a|	(absolute value, |a| < p)	RAM@0x470	EOS
	Curve modulus value *p*	(odd integer prime, $0< p < 2^{640}$)	RAM@0x4D0	
	Curve base point G coordinate x	($x < p$)	RAM@0x678	
	Curve base point G coordinate y	($y < p$)	RAM@0x6D0	
	Public-key curve point Q coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point Q coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part *r*	($0 < r < n$)	RAM@0x10E0	
	Signature part *s*	($0 < s < n$)	RAM@0xC68	
	Hash of message *z*	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order *n*	(integer prime)	RAM@0x1088	

1. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.

Table 373. ECDSA verification - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– (0 < *r* < n)	RAM@0x578	EOS

36.5.18 ECC complete addition

ECC complete addition computes the addition of two given points on an elliptic curve.

Operation instructions are summarized in *Table 374*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x,y) conversion (X,Y,Z) = (x,y,1) can be used. To convert resulting point to Jacobian coordinates conversion (x,y)= (X/Z^2,Y/Z^3) can be used.

Table 374. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	First point P coordinate X	(x < *p*)	RAM@0x628	
	First point P coordinate Y	(y < *p*)	RAM@0x680	
	First point P coordinate Z	(z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(x < *p*)	RAM@0x730	
	Second point Q coordinate Y	(y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(z < *p*)	RAM@0x7E0	
OUT	Result coordinate X	(x < *p*)	RAM@0xD60	
	Result coordinate Y	(y < *p*)	RAM@0xDB8	
	Result coordinate Z	(z < *p*)	RAM@0xE10	

36.5.19 ECC double base ladder

ECC double base ladder operation consists in the computation of k*P+m*Q, where (P,Q) are two points on an elliptic curve and (k,m) are two scalars. Operation instructions are summarized in *Table 375*.

If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 375. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer k	($0 < k < 2^{640}$)	RAM@0x520	
	Integer m	($0 < m < 2^{640}$)	RAM@0x578	
	First point P coordinate X	(x < *p*)	RAM@0x628	
	First point P coordinate Y	(y < *p*)	RAM@0x680	
	First point P coordinate Z	(z < *p*)	RAM@0x6D8	
	Second point Q coordinate X	(x < *p*)	RAM@0x730	
	Second point Q coordinate Y	(y < *p*)	RAM@0x788	
	Second point Q coordinate Z	(z < *p*)	RAM@0x7E0	
OUT	Result coordinate x	(x < *p*)	RAM@0x578	EOS
	Result coordinate y	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

36.5.20 ECC projective to affine

ECC projective to affine operation computes the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (X, Y, Z), it computes the affine coordinates (x, y) = (X/Z, Y/Z).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 376*.

Table 376. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value p	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate X (projective)	(x < *p*)	RAM@0xD60	
	Point P coordinate Y (projective)	(y < *p*)	RAM@0xDB8	
	Point P coordinate Z (projective)	(z < *p*)	RAM@0xE10	
	Montgomery parameter R2 mod n	-	RAM@0x4C8	
OUT	Point P coordinate x (affine)	(x < *p*)	RAM@0x578	
	Point P coordinate y (affine)	(y < *p*)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

36.6 Example of configurations and processing times

36.6.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b \pmod{p}$.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field GF(*p*)
- the (usually prime) integer *n*, the order of the group generated by G, defined below
- the base point of the curve G, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field GF(*p*), for example **p-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 377 summarizes the family of curves supported by PKA for ECC operations.

Table 377. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

36.6.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 378. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	124600	491000	684000	1133200
	Fast	22700	82000	178000	311000
17	Normal	135700	531400	772400	1288000
	Fast	33800	122500	266500	465800
2^{16} + 1	Normal	180000	693700	1126200	1907200
	Fast	78200	284700	620400	1085000
1024	Protected	9958000	-	-	-
	Normal	5850000	-	-	-
	Fast	5748000	-	-	-
	CRT(1)	1775000	-	-	-
2048	Protected	-	63886000	-	-
	Normal	-	42240000	-	-
	Fast	-	41832000	-	-
	CRT(1)	-	11670000	-	-
3072	Protected	-	-	199403000	-
	Normal	-	-	136830000	-
	Fast	-	-	136325000	-
	CRT(1)	-	-	36886000	-
4096	Protected	-	-	-	454318000
	Normal	-	-	-	316000000
	Fast	-	-	-	315226000
	CRT(1)	-	-	-	84577000

1. CRT stands for chinese remainder theorem optimization (MODE bitfield= 0x07).

Table 379. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	1590000	3083000	5339000	8518000	17818000	21053000	31826000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 380. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	1500000	2744000	4579000	7184000	14455000	16685000	24965000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 381. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
1011000	1495000	2938000	5014000	7979000	16804000	19254000	29582000

Table 382. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
967000	1419000	2768000	4784000	7547000	15854000	18257000	28257000

Table 383. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
47600	78000	148300	253000	419000	838400	1049300

Table 384. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
10000	12000	18000	26000	39000	53000	89000

Table 385. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
3400	4200	6100	8300	10900	17200	-	-

Table 386. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
8600	8710	11870	17000	102000	410000	506000	822000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

36.7 PKA interrupts

There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 36.4.7*
2. Access to unmapped address (ADDRERRF), see *Section 36.4.7*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 36.4.7*
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 387 gives a summary of the available features.

Table 387. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

36.8 PKA registers

36.8.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication (protected)
100100: ECDSA sign (protected)
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE bit is ignored.

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

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA

1: Enable PKA. PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

36.8.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

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

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE bitfield can be used.

1: Only ECDSA verification (MODE = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK
This bit is asserted when PKA initialization is complete.
0: PKA is not initialized correctly. START bit cannot be set.
1: PKA is initialized correctly and can be used normally.

Note: When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

36.8.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

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

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

Bit 21 **OPERRFC**: Clear operation error flag
0: No action
1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag
0: No action
1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
0: No action
1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

36.8.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 36.4.3: PKA reset and clocks.

36.8.5 PKA register map

Table 388. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value											0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
	Reset value											0	0	0		0	0															0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0		0																	

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

37 On-the-fly decryption engine (OTFDEC)

This section only applies to STM32H533/73 devices.

37.1 OTFDEC introduction

OTFDEC allows on-the-fly decryption of the AHB traffic based on the read request address information. Four independent and non-overlapping encrypted regions can be defined in OTFDEC.

OTFDEC uses AES-128 in counter mode to achieve the lowest possible latency. As a consequence, each time the content of one encrypted region is changed, the entire region must be re-encrypted with a different cryptographic context (key or initialization vector). This constraint makes OTFDEC suitable to decrypt read-only data or code, stored in external NOR flash.

Note: When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to access the flash memory using the memory-mapped mode of the flash memory controller.

When security is enabled in the product, OTFDEC can be programmed only by a secure host.

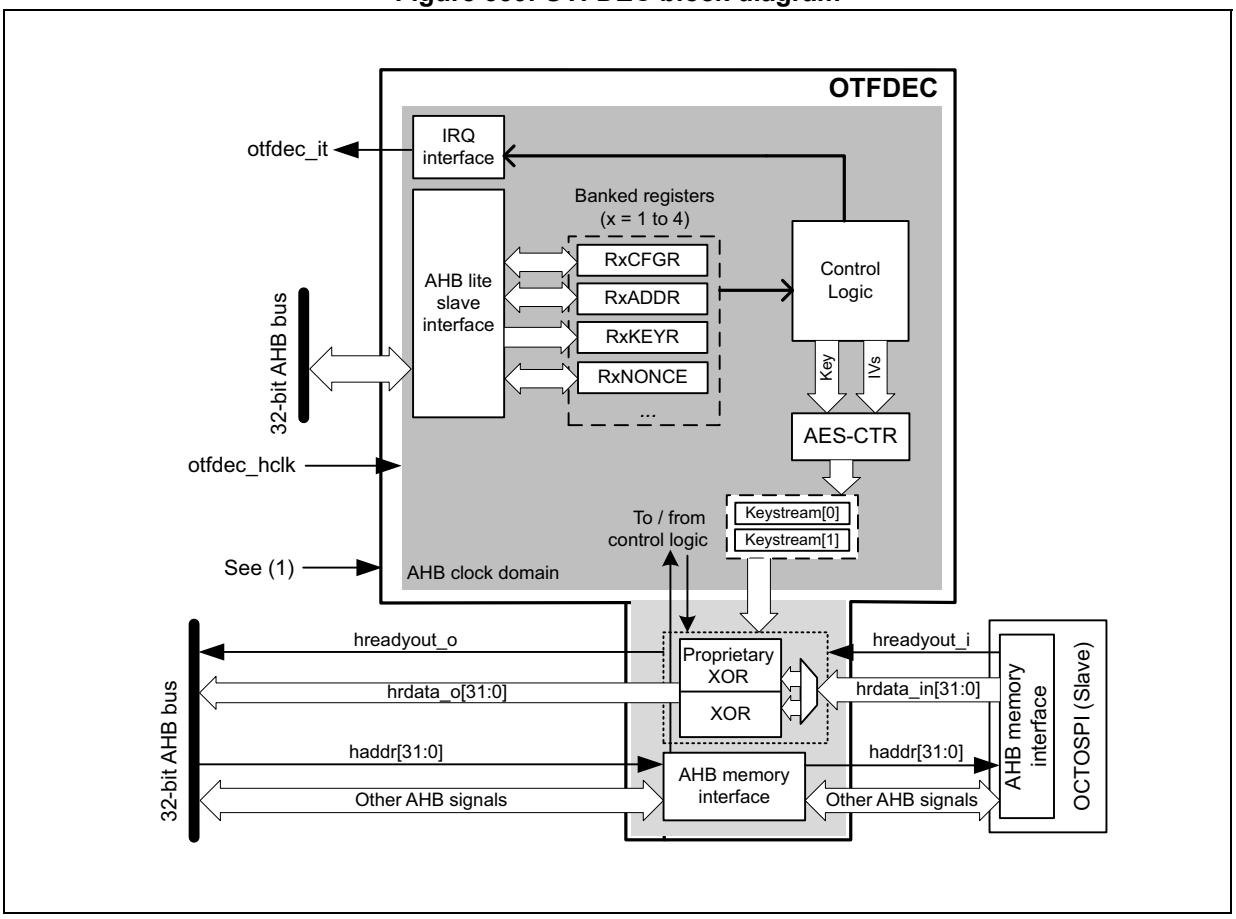
37.2 OTFDEC main features

- On-the-fly 128-bit decryption during the OCTOSPI Memory-mapped read operations (single or multiple).
  - Use of AES in counter (CTR) mode, with two 128-bit keystream buffers
  - Support for any read size
  - Physical address of the reads used for the encryption/decryption
- Up to four independent encrypted regions
  - Granularity of the region definition: 4096 bytes
  - Region configuration write-locking mechanism
  - Each region has its own 128-bit key, two bytes firmware version, and eight bytes application-defined nonce. At least one of those must be changed each time an encryption is performed by the application.
- Encryption keys confidentiality and integrity protection
  - Write-only registers, with software locking mechanism
  - Availability of 8-bit CRC as public key information
- Support for OCTOSPI pre-fetching mechanism
- Possibility to select an enhanced encryption mode to add a proprietary layer of protection on top of AES stream cipher (execute only)
- Privileged-aware AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- Secure only programming if TrustZone security is enabled in the product
- Encryption mode

37.3 OTFDEC functional description

37.3.1 OTFDEC block diagram

Figure 339. OTFDEC block diagram



1. otfdec_tzen

37.3.2 OTFDEC internal signals

Table 389 describes a list of useful to know internal signals available at OTFDEC level, not at the product level (on pads).

Table 389. OTFDEC internal input/output signals

Signal name	Signal type	Description
otfdec_hclk	Digital input	AHB bus clock
otfdec_it	Digital output	OTFDEC global interrupt request
otfdec_tzen	Digital input	OTFDEC TrustZone enable, controlling TrustZone features of the peripheral (TZEN)

The TZEN option bit in FLASH is used to activate TrustZone in the device.

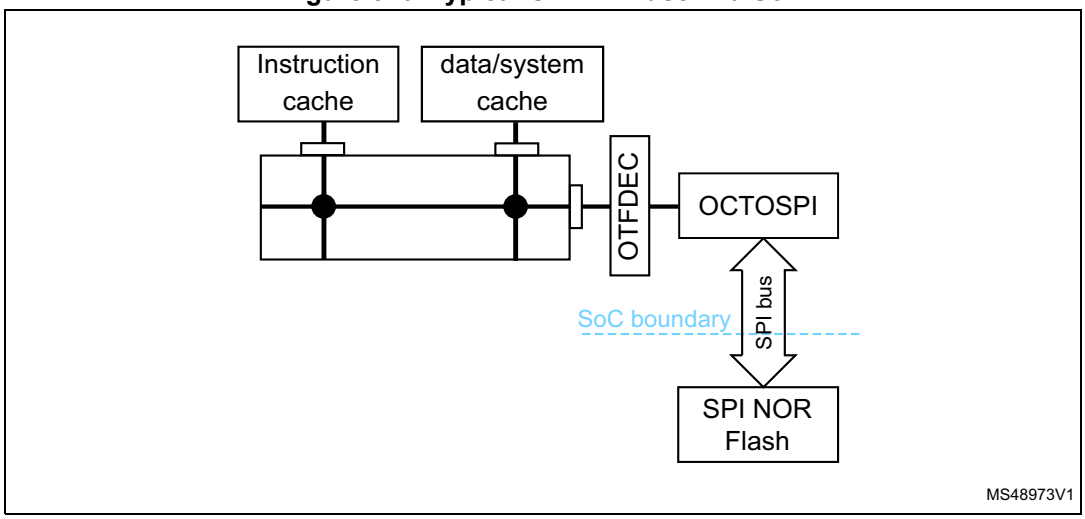
- TZEN = 1: TrustZone security is enabled in the product.
- TZEN = 0: TrustZone security is disabled in the product.

37.3.3 OTFDEC on-the-fly decryption

Introduction

Typical usage for OTFDEC is shown on *Figure 340*.

Figure 340. Typical OTFDEC use in a SoC



Original purpose of OTFDEC is to protect the confidentiality of read-only firmware libraries stored in external SPI NOR flash devices.

A special locking scheme is available in OTFDEC in order to protect the integrity of the decryption keys and also to protect the other configurations against software denial of services attacks. OTFDEC access to most registers can be made privileged-only by setting PRIV bit in OTFDEC_PRIVCFGR register. OTFDEC is only writeable by TrustZone CPU, when TrustZone security is activated.

When OTFDEC is used in conjunction with OCTOSPI, it is mandatory to read the flash memory using the Memory-mapped mode of the flash controller.

On top of decrypting on-the-fly, OTFDEC can also encrypt 32-bit word at a time (see *Section 37.5.3: Encrypting for OTFDEC* for more details).

OTFDEC architecture

OTFDEC analyzes all AHB read transfers on the associated AHB bus. If the read request is within one of the four regions programmed in OTFDEC, the control logic triggers a keystream computation based on AES algorithm in counter mode. This keystream is then used to decrypt on-the-fly the data present in the read transfer from the OCTOSPI AHB master, tying low the HREADYOUT signal of this master while the keystream information is being computed (this takes up to 11 cycles). Any accesses outside the enabled OTFDEC regions belong to a non-encrypted region.

Each OTFDEC region is programmed through OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR, OTFDEC_RxNONCER and

OTFDEC_RxKEYR registers, where x = 1 to 4. In OTFDEC_RxCFGR, the MODE bits define the OTFDEC operating mode (standard or enhanced encryption).

Granularity for the region determination is 4096 bytes.

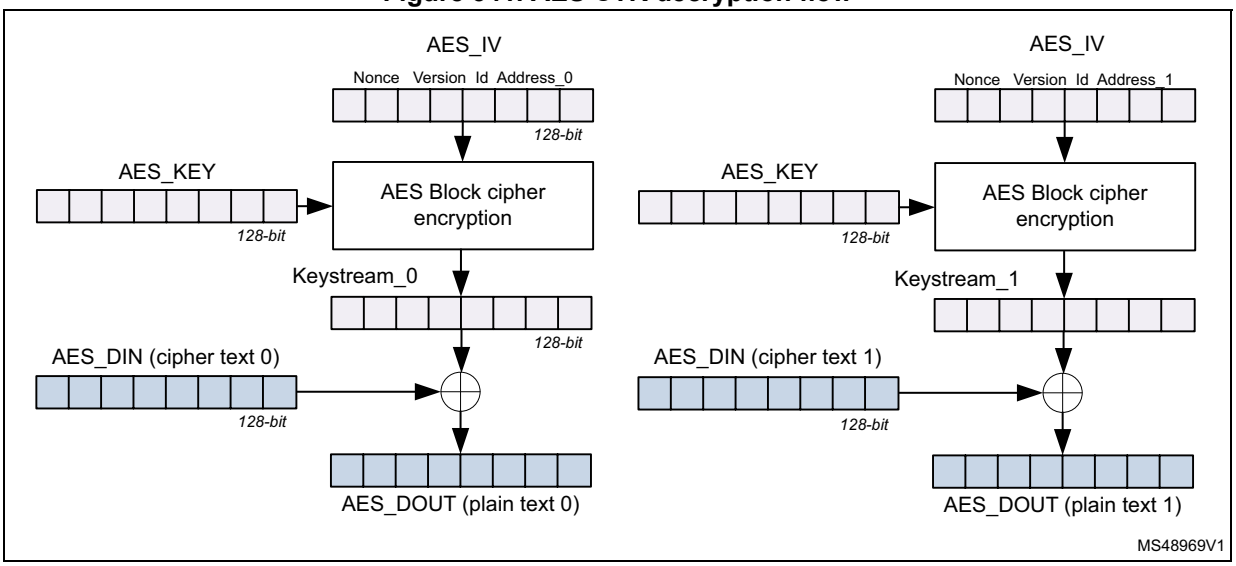
Note: Although OTFDEC does not prevent region overlapping, it is not a valid programming and it must be avoided by application software.

OTFDEC can decrypt incremental or wrap bursts only if they do not cross the 4096-byte aligned address boundaries.

37.3.4 OTFDEC usage of AES in counter mode decryption

Figure 341 shows how OTFDEC uses industry standard Advanced Encryption Standard (AES) algorithm in counter chaining mode. This mode is specified by NIST in *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation*.

Figure 341. AES CTR decryption flow



Every 128-bit data block, a special keystream information is computed using AES block cipher, as defined below:

- initialization vector AES_IV[127:0] = RxNONCER1[31:0] || RxNONCER0[31:0] || 0b0000 0000 0000 0000 || RxCFGR[31:16] || 0b00 || (x-1) || ReadAddress[31:4]
- key material AES_KEY[127:0] = R*x*KEYR3[31:0] || R*x*KEYR2[31:0] || R*x*KEYR1[31:0] || R*x*KEYR0[31:0]

Note: Above x is the RegionID of the selected encrypted region (x=1 to 4).
ReadAddress is the AHB address of the encrypted data block, modulo 128-bit.

Resulting 128-bit keystream is XORed with 128-bit cipher text data to produce the 128-bit clear text data.

- AES_DIN and AES_DOUT data blocks are constructed following the rule below (“|” represents a binary concatenation):
  AES_Dx[127:0]= AHB_word(@ | 0xC)[31:0] | AHB_word(@ | 0x8)[31:0] | AHB_word(@ | 0x4)[31:0] || AHB_word(@ | 0x0)[31:0], where @ is the hexadecimal address used to compute the keystream (ReadAddress[31:4] above).

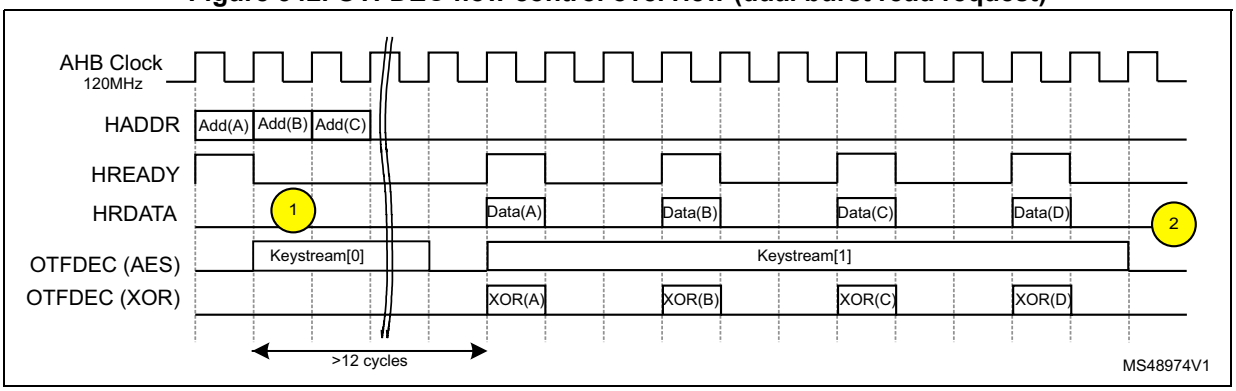
When the read request is not within an encrypted region, or the decryption is not enabled in this region, the AHB data is not changed.

Note: When the application sets the MODE bitfield to 11 in OTFDEC_RxCFGR, an additional layer of protection is added on top of the AES stream cipher. This enhanced encryption mode can only be used with instructions (execute-only region).

37.3.5 Flow control management

Figure 342 shows how OTFDEC manages one INCR4 AHB burst that corresponds to one 128-bit AES data block.

Figure 342. OTFDEC flow control overview (dual burst read request)



with the following notes:

1. OTFDEC enforces HREADY signal from the AHB master low as it is not ready to decrypt data (keystream computation).
2. Thanks to the keystream buffer, OTFDEC can be ready to process a new batch of data within 12 cycles in this configuration (120 MHz AHB clock, 104 MHz SPI bus delivering 2 bytes per SPI clock).

37.3.6 OTFDEC error management

OTFDEC automatically manages errors defined as below:

- Illegal read to OTFDEC_RxKEYR registers
- Illegal write to OTFDEC_RxKEYR registers while CONFIGLOCK or KEYLOCK = 1 in OTFDEC_RxCFGR, while the access is secure. If the security is disabled in the product, the same error occurs when the access is nonsecure.
- Illegal write to OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR or OTFDEC_RxNONCER registers while CONFIGLOCK = 1 in OTFDEC_RxCFGR (x = 1 to 4), while the access is secure. If the security is disabled in the product the same error occurs when the access is nonsecure.
- Illegal read to an execute-only region (MODE[1:0] = 11). Such illegal request returns 0x0, without bus error.
- Execution request to a region while encryption is enabled (ENC = 1). The request returns 0x0, without bus error.
- Key error: read request to an encrypted region while its key registers are null or not properly initialized (KEYCRC = 0x0). Source of the error can be an incorrect key loading sequence (see KEYCRC in OTFDEC_RxCFGR) or it can be an abort event

(tamper detection, unauthorized debug connection, untrusted boot, RDP level regression). Such read request returns 0x0, without bus error.

- Write to any registers while the access is nonsecure, if TrustZone security is enabled in the product.

  This last error is managed and cleared through TrustZone interrupt controller, as described in the GTZC section of the product reference manual.

For these errors (except the last one), an interrupt can be generated if the SEIE, XONEIE or KEIE bit is set in OTFDEC_IER register (see *Section 37.4*).

Note: After a key error, OTFDEC keys must be properly initialized again, and a reset of OTFDEC may be needed if registers are locked.

37.4 OTFDEC interrupts

There are three independent maskable interrupt sources generated by the OTFDEC, signaling following security events:

- Illegal read or write access to keys (SEIF flag), see *Section 37.3.6*
- Illegal write to a region configuration while CONFIGLOCK = 1 (SEIF flag), see *Section 37.3.6*
- Read access to an execute-only region (MODE[1:0] = 11), triggering the XONEIF flag
- Executing while encryption is enabled (XONEIF flag)
- Key error (encrypted regions read as zero) triggering the KEIF flag, see *Section 37.3.6*.

Interrupt sources are connected to the same global interrupt request signal.

OTFDEC interrupt sources can be enabled/disabled by setting the corresponding SEIE, XONEIE or KEIE bit in OTFDEC_IER, as described in *Table 390*. Status of the interrupt event is found in OTFDEC_ISR, and this event can be cleared using OTFDEC_ICR.

Table 390. OTFDEC interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
OTFDEC	Security error	SEIF	SEIE	Set SEIF in OTFDEC_ICR
	Execute-only Execute while encryption	XONEIF	XONEIE	Set XONEIF in OTFDEC_ICR
	Key error	KEIF	KEIE	Set KEIF in OTFDEC_ICR

1. The event flags are found in the OTFDEC_ISR register.

37.5 OTFDEC application information

37.5.1 OTFDEC initialization process

Introduction

One key aspect of OTFDEC is the trusted initialization of its registers, as it involves secret keys. Two trusted initialization schemes are recommended here below.

Note: Those sequences are for production code, as during firmware development, it is not always recommended to lock the key or the region configuration.

Writes to configuration registers are effective when the configuration locks allow it, even if the region is enabled.

Initialization scheme 1: one key for all regions

In this scheme, one entity owns the secret key used to decrypt the four protected regions. The recommended OTFDEC configuration sequence is described below:

1. For *x* = 1 to 4, write the correct MODE[1:0] value in OTFDEC_RxCFGR.
2. For *x* = 1 to 4, program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
3. For *x* = 1 to 4, check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers in this region *x* are no more writable).
4. To do to decrypt a region x (task that does not necessarily have to be performed by the entity that owns the decryption keys):
   a) Verify if the key CRC corresponds to the encrypted binary stored in the region.
   b) Fill the detailed information corresponding to this binary (nonce, start address, end address, version number).
   c) Enable decryption of this region using REG_EN.
   d) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence, step 1 above must be done before step 2, and MODE bits must not be modified after step 2.

Initialization scheme 2: one key per region

In this scheme, one entity can own the secret used to decrypt one (or more) protected region. The recommended OTFDEC configuration sequence is described below:

1. To do to decrypt a region x (this task **must** be performed by the entity that owns the corresponding key):
   a) Write the correct MODE[1:0] value in OTFDEC_R*x*CFGR.
   b) Program OTFDEC_RxKEYR registers using the sequence described in KEYCRC (to have a valid CRC). Warning as key registers are write only.
   c) Check the key CRC. If OK, set KEYLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (key registers are no more writable).
   d) Fill the detailed information corresponding to the protected firmware (nonce, start address, end address, version number).
   e) Enable decryption of this region using REG_EN.
   f) Set CONFIGLOCK bit in OTFDEC_R*x*CFGR. This bit cannot be cleared (the region configuration is no more writable).

Caution: For a given region, when MODE bits are changed, the key registers and associated CRC are cleared by hardware. As a consequence step a) above must be done before step b), and MODE bits must not be modified after step b).

37.5.2 OTFDEC and power management

Each time OTFDEC is reset, the correct key loading sequence described in *Section 37.5.1* must be performed (in this case KEYCRC = 0 in OTFDEC_RxCFGR).

It is recommended for application software to verify this point each time OTFDEC is reset by hardware.

37.5.3 Encrypting for OTFDEC

Code and data standard encryption

OTFDEC uses standard AES in counter mode when processing a binary stored in a protected region with MODE[1:0] = 10. When this mode is selected, any AES compatible hardware accelerator or library can be used to encrypt those protected libraries. OTFDEC can be used as well, as described in enhanced encryption section below (with MODE[1:0] = 10).

Definition and endianness of the AES inputs and outputs are defined in *Section 37.3.4: OTFDEC usage of AES in counter mode decryption*.

Enhanced encryption with OTFDEC

OTFDEC uses a proprietary layer of protection on top of the standard AES in counter mode when processing a code stored in a protected region with MODE[1:0] = 11.

Enhanced encryption mode can be used to increase the robustness against tampering.

Recommended sequence to encrypt using OTFDEC is described below:

1. The application in charge of the encryption sets the ENC bit in OTFDEC_CR. This application must run in TrustZone secure mode when TrustZone security is enabled in the product. If PRIV bit is set in OTFDEC_PRIVCFGR, this application must be privileged.
2. Encryption application initializes OTFDEC as described in *Section 37.5.1: OTFDEC initialization process*. OCTOSPI must also be properly clocked, so that OTFDEC is fully functional in encryption mode. This step can also be done before step 1.
3. Encryption application writes 32-bit of clear-text data at the expected protected address, then reads it back encrypted at the same address to store it in RAM. Note that this data stays inside the device, as it is intercepted by OTFDEC in encryption mode.
4. Encryption application goes back to previous step (changing the address) until the whole binary is processed.
5. Encryption application clears the ENC bit in OTFDEC_CR. Another application can then take the encrypted binary and flash it to the correct address in external flash.

There are few important notes about this procedure:

- Encryption granularity is 32-bit (single 32-bit access is mandatory).
- While ENC bit is set, reads to non-encrypted regions return normal data (such as no encryption nor decryption). While in encryption mode, no access to OCTOSPI (including registers) must be done. This is because the OTFDEC cuts the communication with OCTOPSI while ENC bit is set.
- OTFDEC does not support execution while ENC = 1 (only encrypted data reads). Upon illegal execution detection a XONEIF flag is raised and zero is returned.

37.5.4 OTFDEC key CRC source code

Below is the CRC source code that can be used to compare with the result of the computation provided by OTFDEC in KEYCRC bitfield after loading the keys in OTFDEC_RxKEYR registers.

```
uint8_t getCRC(uint32_t * keyin)
{
  const uint8_t CRC7_POLY = 0x7;
  const uint32_t key_strobe[4] = {0xAA55AA55, 0x3, 0x18, 0xC0};
  uint8_t  i, j, k, crc = 0x0;
  uint32_t  keyval;

  for (j = 0; j < 4; j++)
  {
    keyval = *(keyin+j);
    if (j == 0)
    {
      keyval ^= key_strobe[0];
    }
    else
    {
      keyval ^= (key_strobe[j] << 24) | (crc << 16) | (key_strobe[j] << 8)
| crc;
    }

    for (i = 0, crc = 0; i < 32; i++)
    {
      k = (((crc >> 7) ^ (keyval >> (31-i))&0xF)) & 1;
      crc <<= 1;
      if (k)
      {
        crc ^= CRC7_POLY;
       }
    }
    crc^=0x55;
  }
  return crc;
}
```

37.6 OTFDEC registers

37.6.1 OTFDEC control register (OTFDEC_CR)

Address offset: 0x0

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

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

Bit 0 **ENC**: Encryption mode bit

When this bit is set, OTFDEC is used in encryption mode, during which application can write clear text data then read back encrypted data. When this bit is cleared (default), OTFDEC is used in decryption mode, during which application only read back decrypted data. For both modes, cryptographic context (keys, nonces, firmware versions) must be properly initialized. When this bit is set, only data accesses are allowed (zeros are returned otherwise, and XONEIF is set). When MODE = 11, enhanced encryption mode is automatically selected.

0: OTFDEC working in decryption mode

1: OTFDEC working in encryption mode

Note: When ENC bit is set, no access to OCTOSPI must be done (registers and Memory-mapped region).

37.6.2 OTFDEC privileged access control configuration register (OTFDEC_PRIVCFGR)

Address offset: 0x10

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

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

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

Bit 0 **PRIV:** Privileged access protection.

0: No additional protection is added on OTFDEC register accesses.

1: An additional protection is added when accessing all registers except OTFDEC_PRIVCFGR:

- Unprivileged read accesses to registers return zeros
- Unprivileged write accesses to registers are ignored.

Note: This bit can only be written in privileged mode. There is no limitations on reads.

37.6.3 OTFDEC region x configuration register (OTFDEC_RxCFGR)

Address offset: 0x20 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

Writes are ignored if CONFIGLOCK bit is set to 1.

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

Bits 31:16 **REG_VERSION[15:0]**: region firmware version

This 16-bit bitfield must be correctly initialized before the region corresponding REG_EN bit is set in OTFDEC_RxCFGR.

Bits 15:8 **KEYCRC[7:0]**: region key 8-bit CRC

When KEYLOCK = 0, KEYCRC bitfield is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new computation starts as soon as a new valid sequence is initiated, and KEYCRC is read as zero until a valid sequence is completed.

When KEYLOCK = 1, KEYCRC remains unchanged until the next reset.

CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm $X^8 + X^2 + X + 1$ (according the convention). Source code is available in *Section 37.5.4*.

This field is read only.

Note: CRC information is updated only after the last bit of the key has been written.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE[1:0]**: operating mode

This bitfield selects the OTFDEC operating mode for this region:

10: All read accesses are decrypted (instruction or data).

11: Enhanced encryption mode is activated, and only instruction accesses are decrypted

Others: Reserved

When MODE ≠ 11, the standard AES encryption mode is activated.

When either of the MODE bits are changed, the region key and associated CRC are zeroed.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **KEYLOCK**: region key lock

0: Writes to this region KEYRx registers are allowed.

1: Writes to this region KEYRx registers are ignored until next OTFDEC reset. KEYCRC bitfield is locked.

Note: This bit is set once: if this bit is set, it can only be reset to 0 if the OTFDEC is reset.

Bit 1 **CONFIGLOCK**: region config lock

0: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are allowed.

1: Writes to this region OTFDEC_RxCFGR, OTFDEC_RxSTARTADDR, OTFDEC_RxENDADDR and OTFDEC_RxNONCERy registers are ignored until next OTFDEC reset.

Note: This bit is set once. If this bit is set, it can only be reset to 0 if OTFDEC is reset. Setting this bit forces KEYLOCK bit to 1.

Bit 0 **REG_EN**: region on-the-fly decryption enable

0: On-the-fly decryption is disabled for this region.

1: On-the-fly decryption is enabled for this region. Data are XORed with the corresponding keystream.

Note: Garbage is decrypted if region context (version, key, nonce) is not valid when this bit is set.

37.6.4 OTFDEC region x start address register (OTFDEC_RxSTARTADDR)

Address offset: 0x24 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_START_ADDR[31:0]**: Region AHB start address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR register is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros .

37.6.5 OTFDEC region x end address register (OTFDEC_RxENDADDR)

Address offset: 0x28 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0FFF

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_END_ADDR[31:0]**: Region AHB end address

This register must be written before the region corresponding REG_EN bit in the OTFDEC_RxCFGR register is set, and OTFDEC_RxENDADDR must be strictly greater than OTFDEC_RxSTARTADDR to be valid.

Writing to this register is discarded if performed while the region CONFIGLOCK bit in OTFDEC_RxCFGR is set.

Note: When determining the region the first 12 bits (LSB) and the last 4 bits (MSB) are ignored.

When this register is accessed in read the 4 MSB bits return zeros and the 12 LSB bits return ones.

37.6.6 OTFDEC region x nonce register 0 (OTFDEC_RxNONCER0)

Address offset: 0x2C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_NONCE[31:0]**: Region nonce, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Writing is discarded in this register if performed while the region CONFIGLOCK bit in the OTFDEC_RxCFGR is set.

37.6.7 OTFDEC region x nonce register 1 (OTFDEC_RxNONCER1)

Address offset: 0x30 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_NONCE[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_NONCE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **REG_NONCE[63:32]**: Region nonce, bits [63:32]

Refer to the OTFDEC_RxNONCER0 register for description of the NONCE[63:0] bitfield.

37.6.8 OTFDEC region x key register 0 (OTFDEC_RxKEYR0)

Address offset: 0x34 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

Bits 31:0 **REG_KEY[31:0]**: Region key, bits [31:0]

This register must be written before the region corresponding REG_EN bit in OTFDEC_RxCFGR is set.

Reading this register returns a zero value. Writing to this register is discarded if performed while the region CONFIGLOCK or KEYLOCK bit is set in the OTFDEC_RxCFGR.

Note: When application successfully changes MODE bits in OTFDEC*_RxCFGR and* OTFDEC*_RxKEYR, and associated KEYCRC are erased.*

37.6.9 OTFDEC region x key register 1 (OTFDEC_RxKEYR1)

Address offset: 0x38 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[63:32]**: Region key, bits [63:32]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

37.6.10 OTFDEC region x key register 2 (OTFDEC_RxKEYR2)

Address offset: 0x3C + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[95:64]**: Region key, bits [95:64]
Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

37.6.11 OTFDEC region x key register 3 (OTFDEC_RxKEYR3)

Address offset: 0x40 + 0x30 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REG_KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REG_KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **REG_KEY[127:96]**: Region key, bits [127:96]

Refer to the OTFDEC_RxKEYR0 register for description of the KEY[127:0] bitfield.

37.6.12 OTFDEC interrupt status register (OTFDEC_ISR)

Address offset: 0x300

Reset value: 0x0000 0000

Unprivileged reads return zero if PRIV bit is set in OTFDEC_PRIVCFGR.

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

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

Bit 2 **KEIF:** Key error interrupt flag status

This bit is set by hardware and read only by application. The bit is set when a read access occurs on an encrypted region, while its key registers is null or not properly initialized (KEYCRC = 0x0).

This bit is cleared when the application sets in OTFDEC_ICR the corresponding bit to 1.

0: OTFDEC operates properly.

1: Read access detected on an enabled encrypted region with its key registers null or not properly initialized (KEYCRC = 0x0). OTFDEC returns a zeroed value for the read, and an optional interrupt is generated if bit KEIE is set to 1 in OTFDEC_IER.

After KEIF is set any subsequent read to the region with bad key registers returns a zeroed value. This state remains until those key registers are properly initialized (KEYCRC not zero).

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when a read access and not an instruction fetch is detected on any encrypted region with MODE bits set to 11. Lastly, XONEIF is also set when an execute access is detected while encryption mode is enabled.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No execute-only error status. No interrupt pending.

1: Read access detected on one region with MODE bits set to 11 or execute access detected while ENC = 1. OTFDEC returns a zeroed value for the illegal access, and an optional interrupt is generated if bit XONEIE is set to 1 in OTFDEC_IER.

Bit 0 **SEIF:** Security error interrupt flag status

This bit is set by hardware and read only by application. This bit is set when at least one security error has been detected.

This bit is cleared when application sets in OTFDEC_ICR the corresponding bit to 1.

0: No security error status. No interrupt pending.

1: Security error flag status, with interrupt pending. Actual interrupt generation is dependent on OTFDEC_IER corresponding bit SEIE.

37.6.13 OTFDEC interrupt clear register (OTFDEC_ICR)

Address offset: 0x304

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

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

Bit 2 **KEIF:** Key error interrupt flag clear
This bit is written by application, and always read as 0.
0: KEIF flag status is not affected.
1: KEIF flag status is cleared in OTFDEC_ISR.
Note: Clearing KEIF does not solve the source of the problem (bad key registers). To be able to access again any encrypted region, OTFDEC key registers must be properly initialized again.

Bit 1 **XONEIF:** Execute-only execute-never error interrupt flag clear
This bit is written by application, and always read as 0.
0: XONEIF flag status is not affected.
1: XONEIF flag status is cleared in OTFDEC_ISR.

Bit 0 **SEIF:** Security error interrupt flag clear
This bit is written by application, and always read as 0.
0: SEIF flag status is not affected.
1: SEIF flag status is cleared in OTFDEC_ISR.

37.6.14 OTFDEC interrupt enable register (OTFDEC_IER)

Address offset: 0x308

Reset value: 0x0000 0000

Nonsecure AHB write access (HNONSEC = 1) is discarded if the TrustZone security is enabled in the product.

Unprivileged reads return zero and unprivileged writes are ignored if PRIV bit is set in OTFDEC_PRIVCFGR.

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

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

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

Bit 2 **KEIE:** Key error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when KEIF flag status is set.

0: Interrupt generation on key error flag KEIF is disabled (masked).

1: Interrupt generation on key error flag KEIF is enabled (not masked).

Bit 1 **XONEIE:** Execute-only execute-never error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when XONEIF flag status is set.

0: Interrupt generation on execute-only error XONEIF is disabled (masked).

1: Interrupt generation on execute-only error XONEIF is enabled (not masked).

Bit 0 **SEIE:** Security error interrupt enable

This bit is read and written by application. It controls the OTFDEC interrupt generation when SEIF flag status is set.

0: Interrupt generation on security error SEIF is disabled (masked).

1: Interrupt generation on security error SEIF is enabled (not masked).

37.6.15 OTFDEC register map

Table 391. OTFDEC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	OTFDEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC
	Reset value																																0
0x04-0x0C	Reserved	Reserved																															
0x10	OTFDEC_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV
	Reset value																																0
0x14-0x1C	Reserved	Reserved																															
0x20	OTFDEC_R1CFGR1	REG1_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x24	OTFDEC_R1STARTADDR	REG1_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	OTFDEC_R1ENDADDR	REG1_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	OTFDEC_R1NONCER0	REG1_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	OTFDEC_R1NONCER1	REG1_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	OTFDEC_R1KEYR0	REG1_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 391. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x38	OTFDEC_ R1KEYR1	REG1_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	OTFDEC_ R1KEYR2	REG1_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	OTFDEC_ R1KEYR3	REG1_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44 - 0x4C	Reserved	Reserved																															
0x50	OTFDEC_ R2CFGR	REG2_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x54	OTFDEC_ R2STARTADDR	REG2_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58	OTFDEC_ R2ENDADDR	REG2_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C	OTFDEC_ R2NONCER0	REG2_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60	OTFDEC_ R2NONCER1	REG2_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64	OTFDEC_ R2KEYR0	REG2_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68	OTFDEC_ R2KEYR1	REG2_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C	OTFDEC_ R2KEYR2	REG2_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	OTFDEC_ R2KEYR3	REG2_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74 - 0x7C	Reserved	Reserved																															
0x80	OTFDEC_ R3CFGR	REG3_VERSION[15:0]																KEYCRC[7:0]									Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x84	OTFDEC_ R3STARTADDR	REG3_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x88	OTFDEC_ R3ENDADDR	REG3_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 391. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x8C	OTFDEC_R3NONCER0	REG3_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x90	OTFDEC_R3NONCER1	REG3_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	OTFDEC_R3KEYR0	REG3_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	OTFDEC_R3KEYR1	REG3_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C	OTFDEC_R3KEYR2	REG3_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0	OTFDEC_R3KEYR3	REG3_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4 - 0xAC	Reserved	Reserved																															
0xB0	OTFDEC_R4CFGR	REG4_VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1:0]		KEYLOCK	CONFIGLOCK	REG_EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0xB4	OTFDEC_R4STARTADDR	REG4_START_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB8	OTFDEC_R4ENDADDR	REG4_END_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC	OTFDEC_R4NONCER0	REG4_NONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC0	OTFDEC_R4NONCER1	REG4_NONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC4	OTFDEC_R4KEYR0	REG4_KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC8	OTFDEC_R4KEYR1	REG4_KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC	OTFDEC_R4KEYR2	REG4_KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0	OTFDEC_R4KEYR3	REG4_KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4-0x2FC	Reserved	Reserved																															

Table 391. OTFDEC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x300	OTFDEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x304	OTFDEC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		KEIF	XONEIF	SEIF
	Reset value																														0	0	0
0x308	OTFDEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		KEIE	XONEIE	SEIE
	Reset value																														0	0	0

Refer to *Section 2.3* for the register boundary addresses.

38 Advanced-control timers (TIM1/TIM8)

38.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 38.3.30: Timer synchronization*.

38.2 TIM1/TIM8 main features

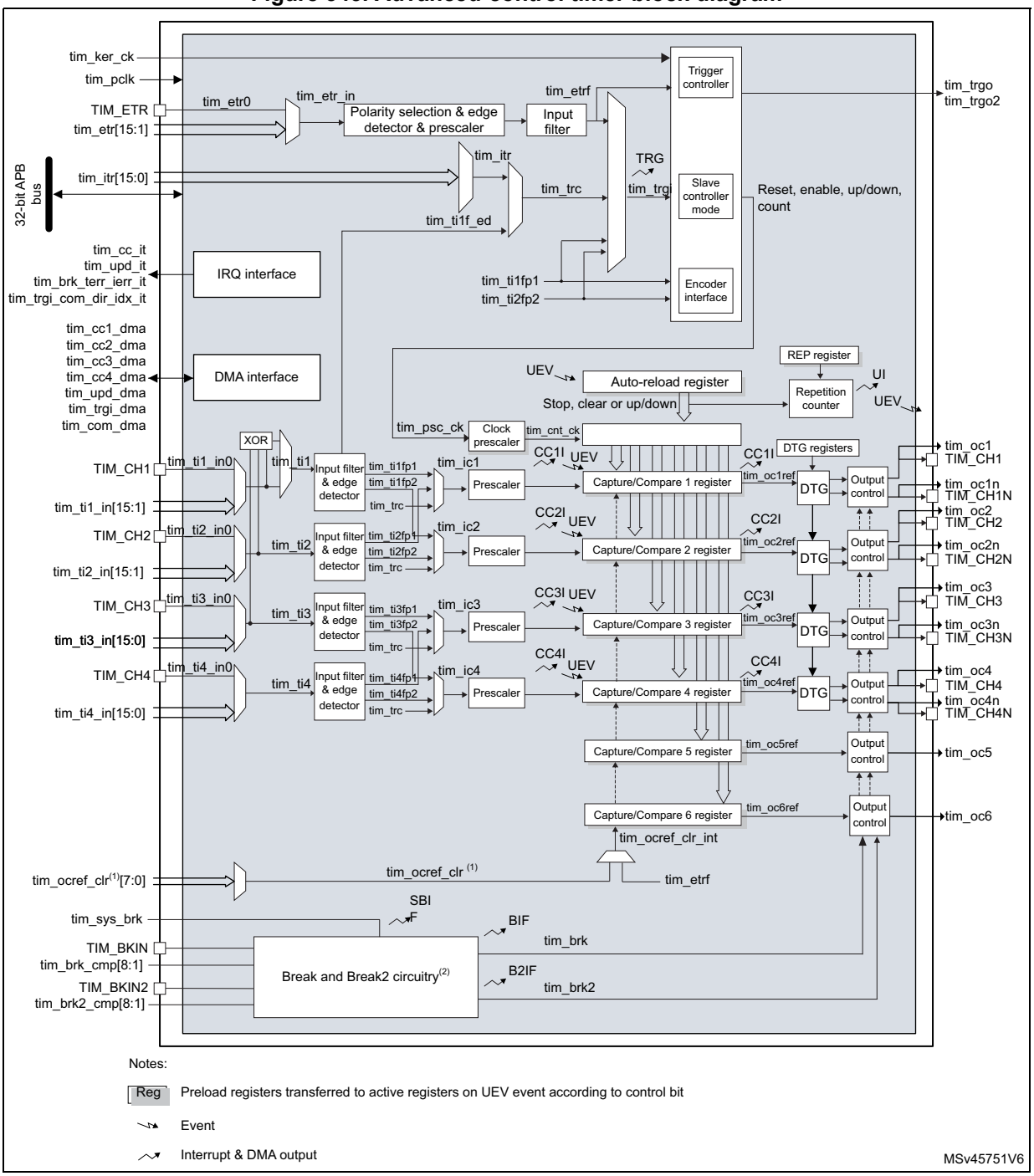
TIM1/TIM8 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Up to six independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

38.3 TIM1/TIM8 functional description

38.3.1 Block diagram

Figure 343. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals*.
2. See *Figure 390: Break and Break2 circuitry overview* for details.

38.3.2 TIM1/TIM8 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 392. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input/output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 393. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 393. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_brk2_cmp[8:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 394, *Table 395*, *Table 396* and *Table 397* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 394. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM1	TIM8
tim_ti1_in0	TIM1_CH1	TIM8_CH1
tim_ti1_in[15:1]	Reserved	Reserved

Table 395. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM1	TIM8
tim_ti2_in0	TIM1_CH2	TIM8_CH2
tim_ti2_in[15:1]	Reserved	Reserved

Table 396. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources	
	TIM1	TIM8
tim_ti3_in0	TIM1_CH3	TIM8_CH3
tim_ti3_in[15:1]	Reserved	Reserved

Table 397. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources	
	TIM1	TIM8
tim_ti4_in0	TIM1_CH4	TIM8_CH4
tim_ti4_in[15:1]	Reserved	Reserved

Table 398 lists the internal sources connected to the tim_itr input multiplexer.

Table 398. Internal trigger connection

Timer internal trigger input signal	TIM1	TIM8
tim_itr0	Reserved	tim1_trgo
tim_itr1	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo

Table 398. Internal trigger connection (continued)

Timer internal trigger input signal	TIM1	TIM8
tim_itr5	tim8_trgo	Reserved
tim_itr6	tim12_trgo	tim12_trgo
tim_itr7	tim13_oc1	tim13_oc1
tim_itr8	tim14_oc1	tim14_oc1
tim_itr9	tim15_trgo	tim15_trgo
tim_itr10	tim16_oc1	tim16_oc1
tim_itr11	tim17_oc1	tim17_oc1
tim_itr[15:12]	Reserved	Reserved

Table 399 lists the internal sources connected to the tim_etr input multiplexer.

Table 399. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM1	TIM8
tim_etr0	TIM1_ETR	TIM8_ETR
tim_etr[2:1]	Reserved	Reserved
tim_etr3	adc1_awd1	adc2_awd1
tim_etr4	adc1_awd2	adc2_awd2
tim_etr5	adc1_awd3	adc2_awd3
tim_etr[15:6]	Reserved	Reserved

Table 400, *Table 401* and *Table 402* list the sources connected to the tim_brk and tim_brk2inputs.

Table 400. Timer break interconnect

tim_brk inputs	TIM1	TIM8
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin
tim_brk_cmp[8:1]	Reserved	Reserved

Table 401. Timer break2 interconnect

tim_brk2 inputs	TIM1	TIM8
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin
tim_brk2_cmp[8:1]	Reserved	Reserved

Table 402. System break interconnect

tim_sys_brk inputs	TIM1/TIM8	Enable bit in SBS_CFGR2 register
tim_sys_brk0	Flash memory double ECC error	ECCL
tim_sys_brk1	Programmable Voltage Detector (PVD)	PVDL
tim_sys_brk2	SRAM double ECC error	SEL
tim_sys_brk3	Cortex-M33 LOCKUP	CLL
CSS	Clock Security System	None (always enabled)

38.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 344 and *Figure 345* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 344. Counter timing diagram with prescaler division change from 1 to 2

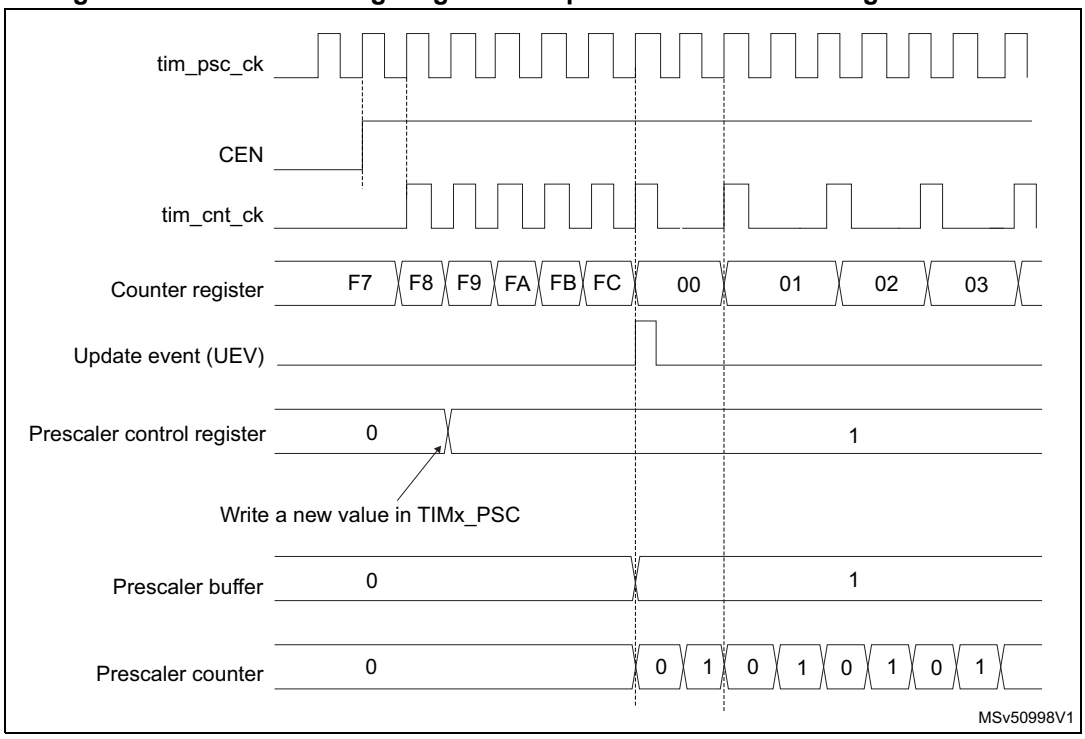
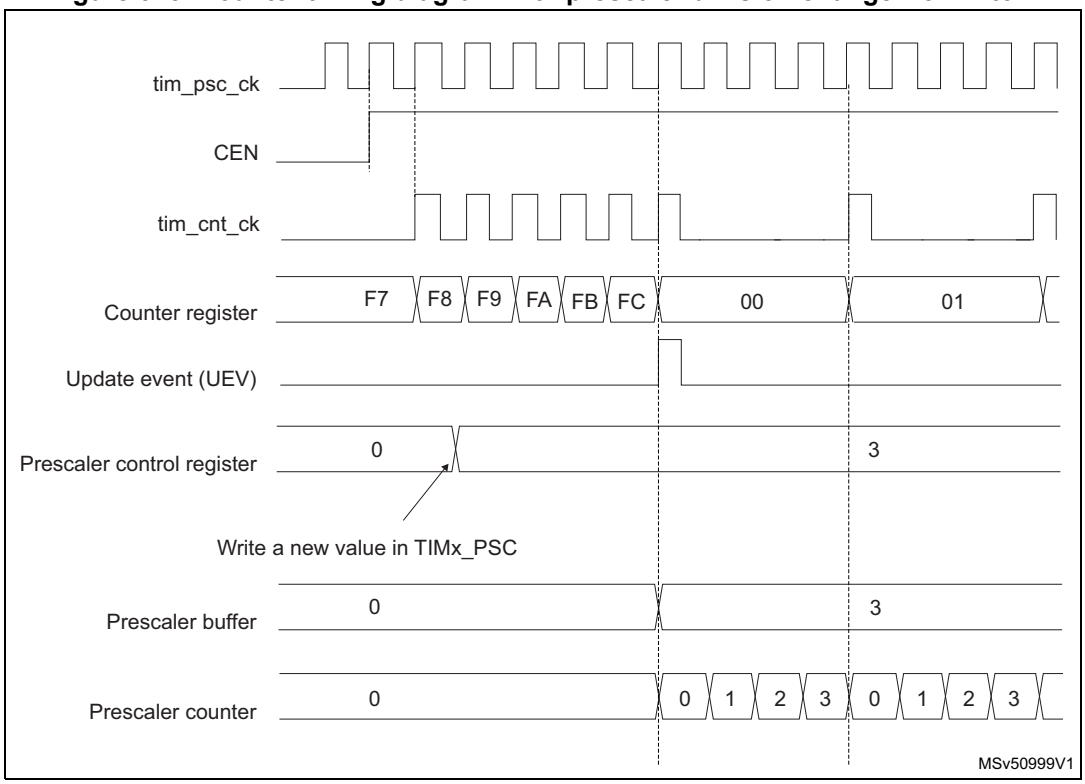


Figure 345. Counter timing diagram with prescaler division change from 1 to 4



38.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 346. Counter timing diagram, internal clock divided by 1

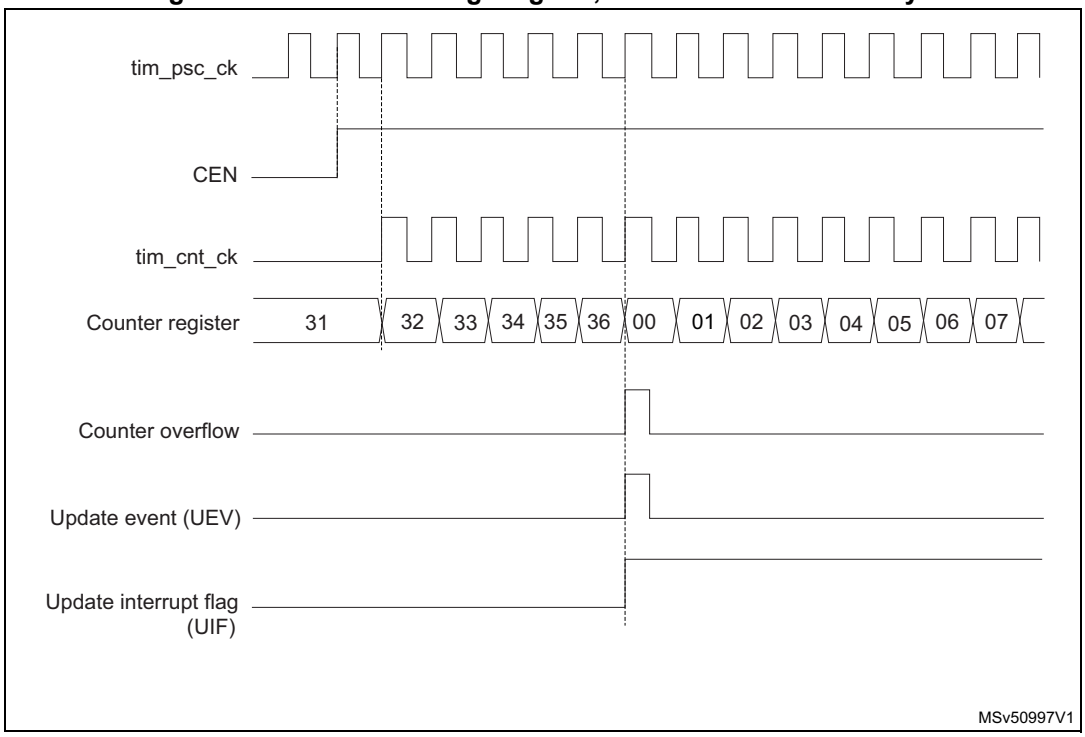


Figure 347. Counter timing diagram, internal clock divided by 2

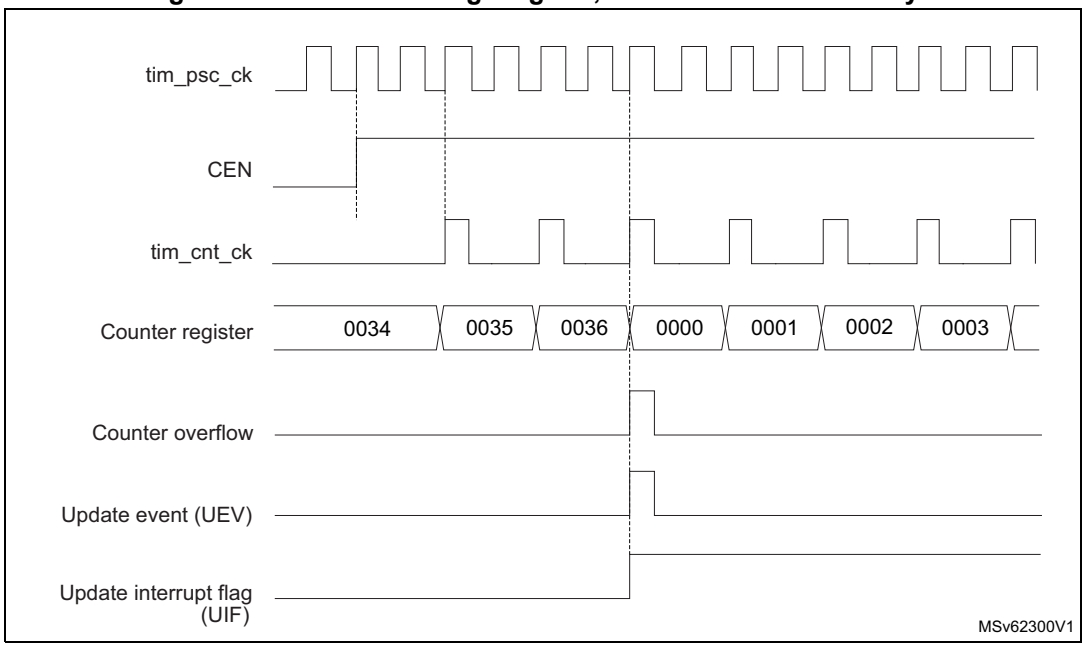


Figure 348. Counter timing diagram, internal clock divided by 4

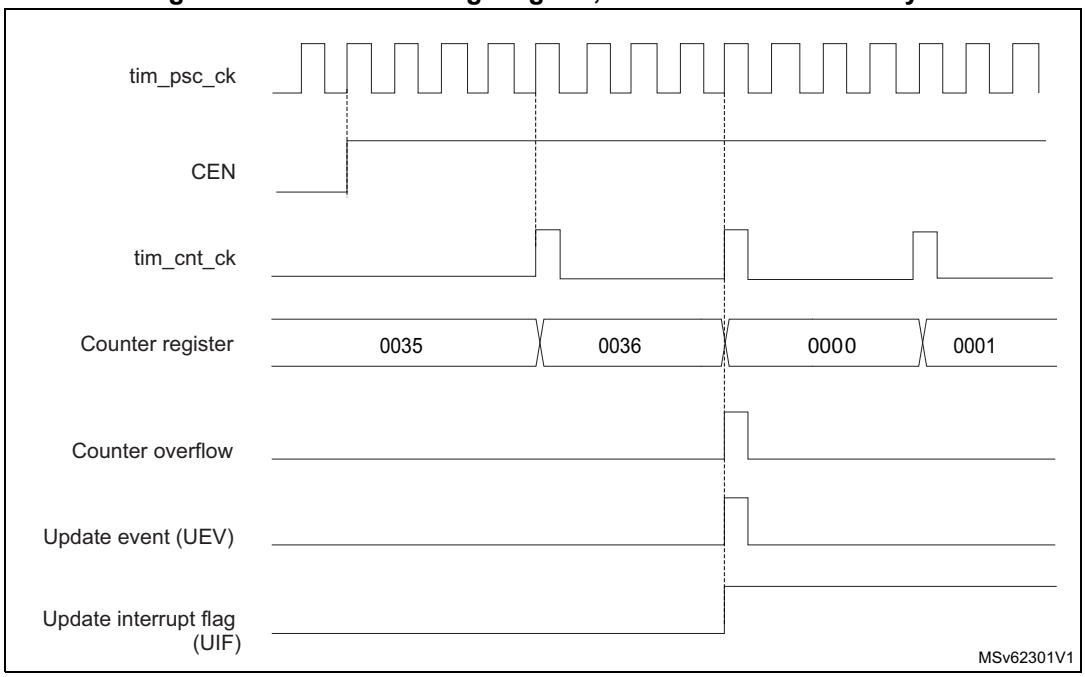


Figure 349. Counter timing diagram, internal clock divided by N

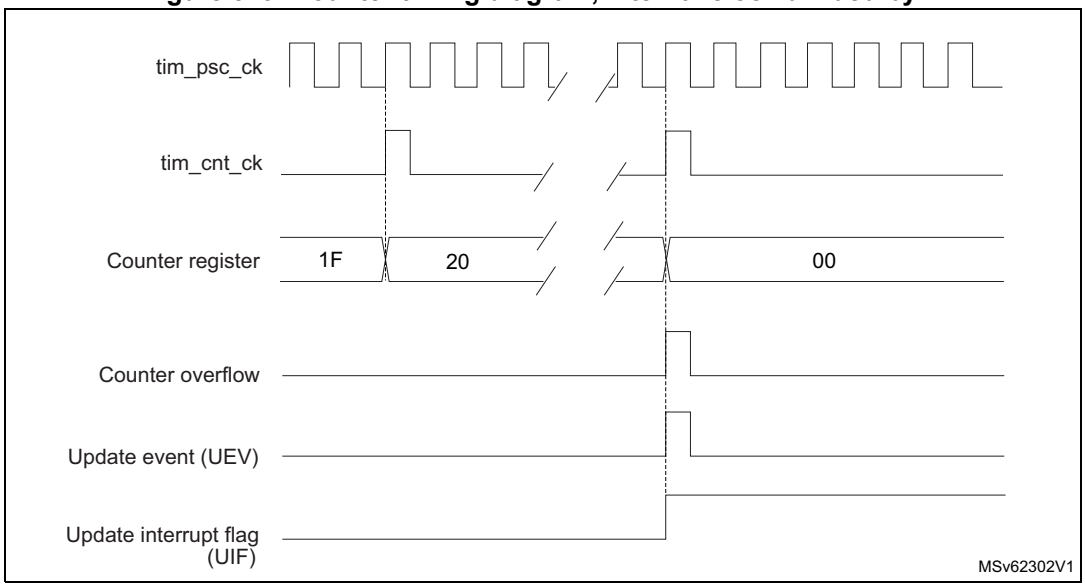


Figure 350. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

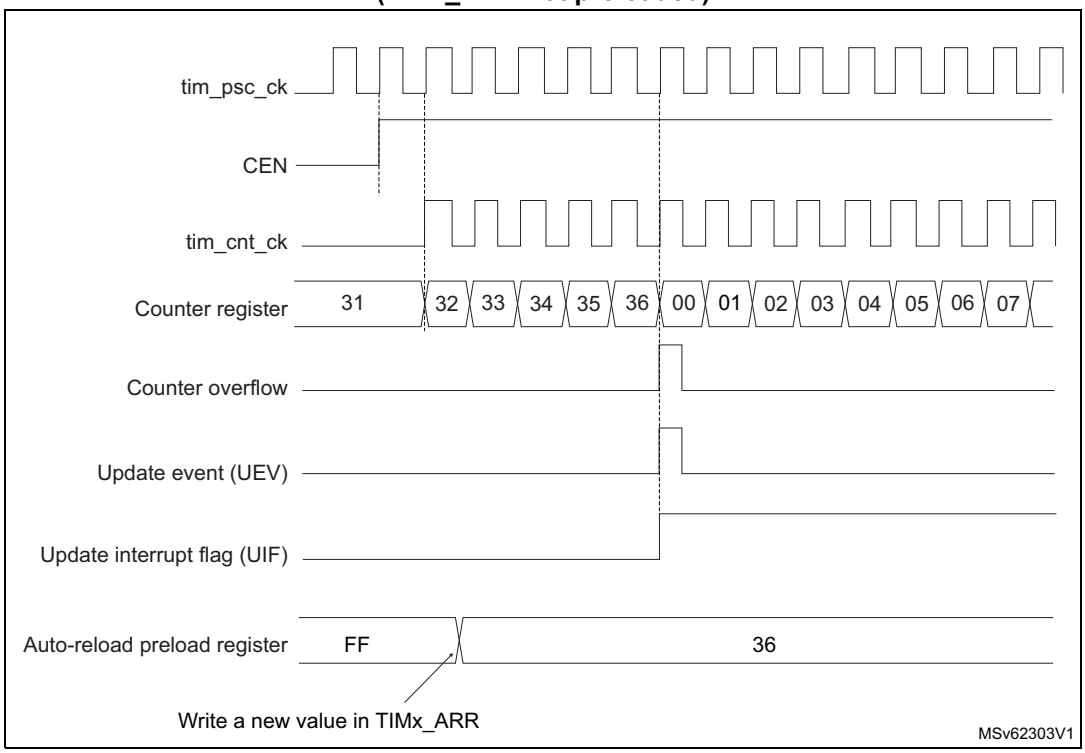
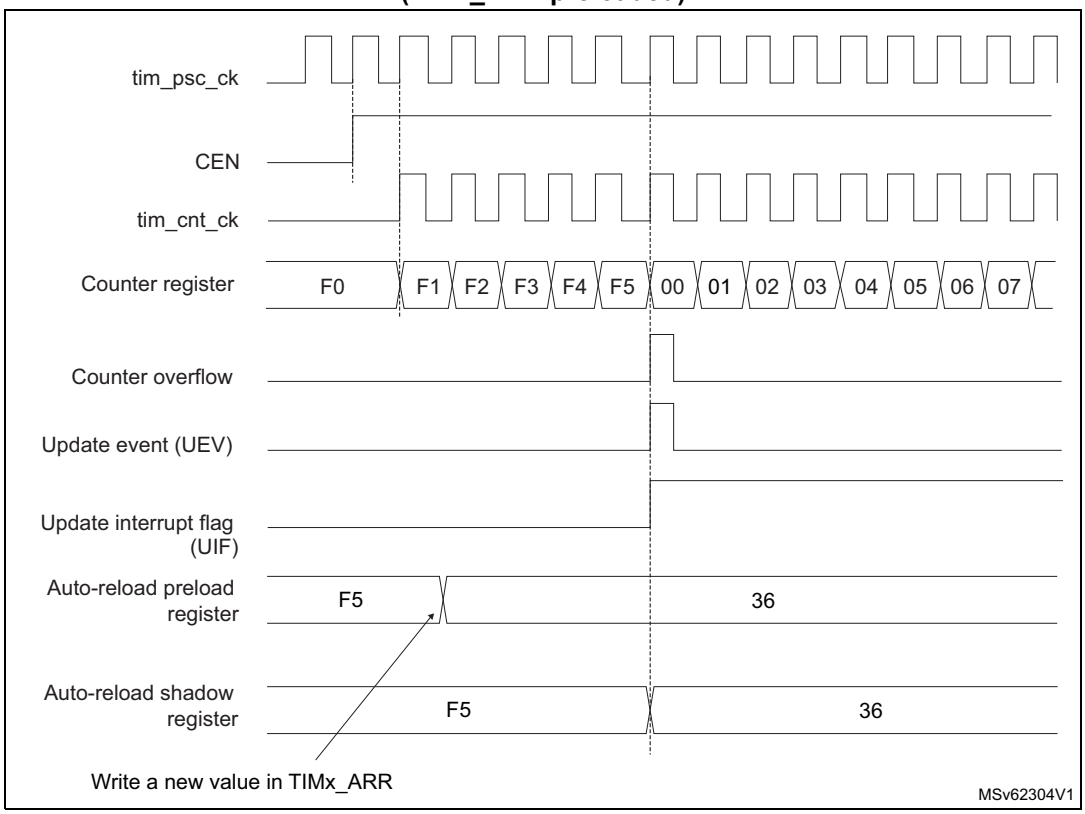


Figure 351. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 352. Counter timing diagram, internal clock divided by 1

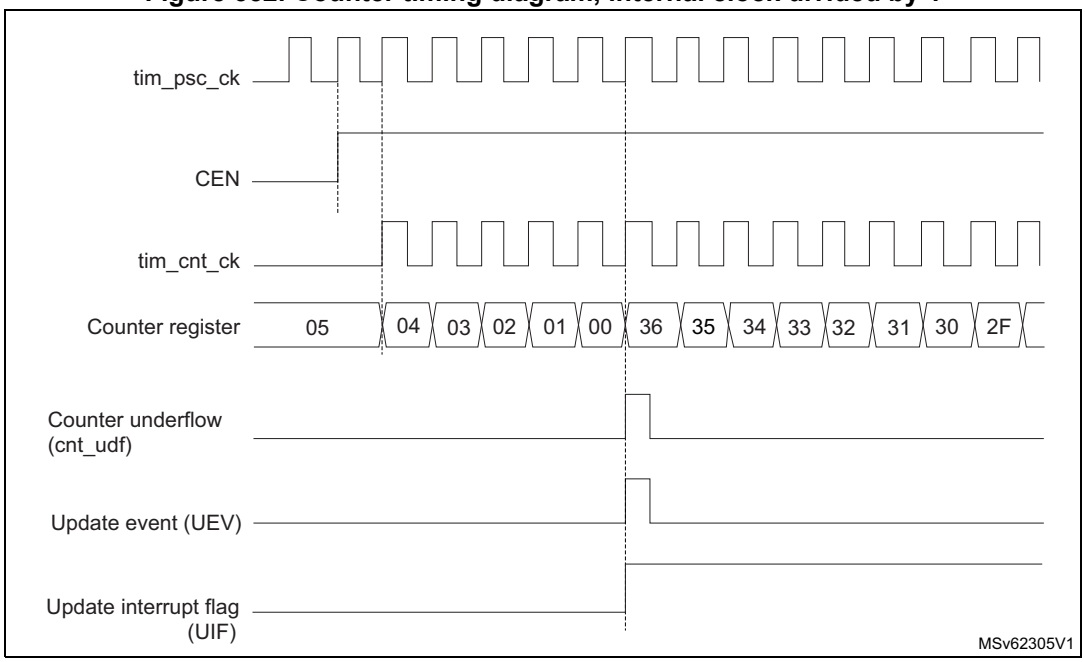


Figure 353. Counter timing diagram, internal clock divided by 2

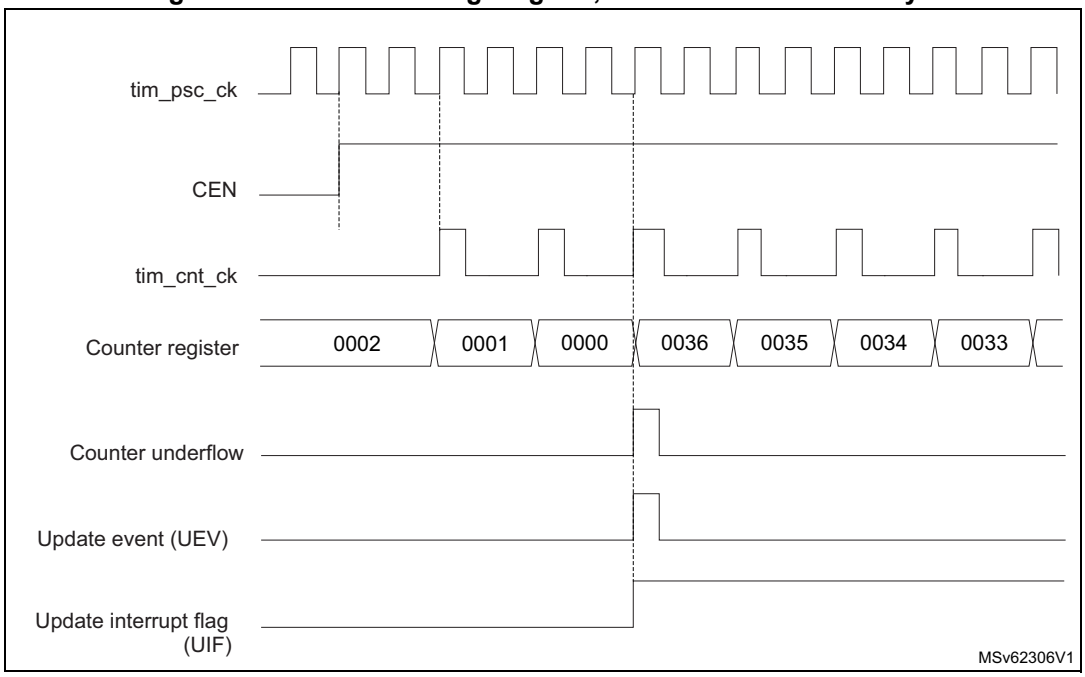


Figure 354. Counter timing diagram, internal clock divided by 4

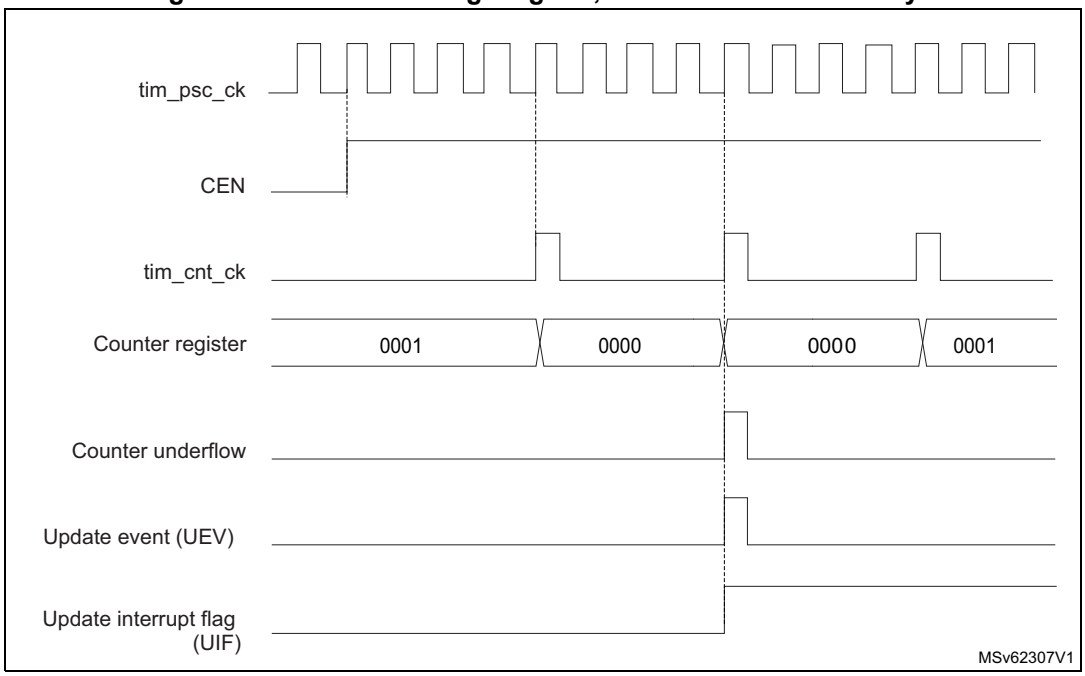


Figure 355. Counter timing diagram, internal clock divided by N

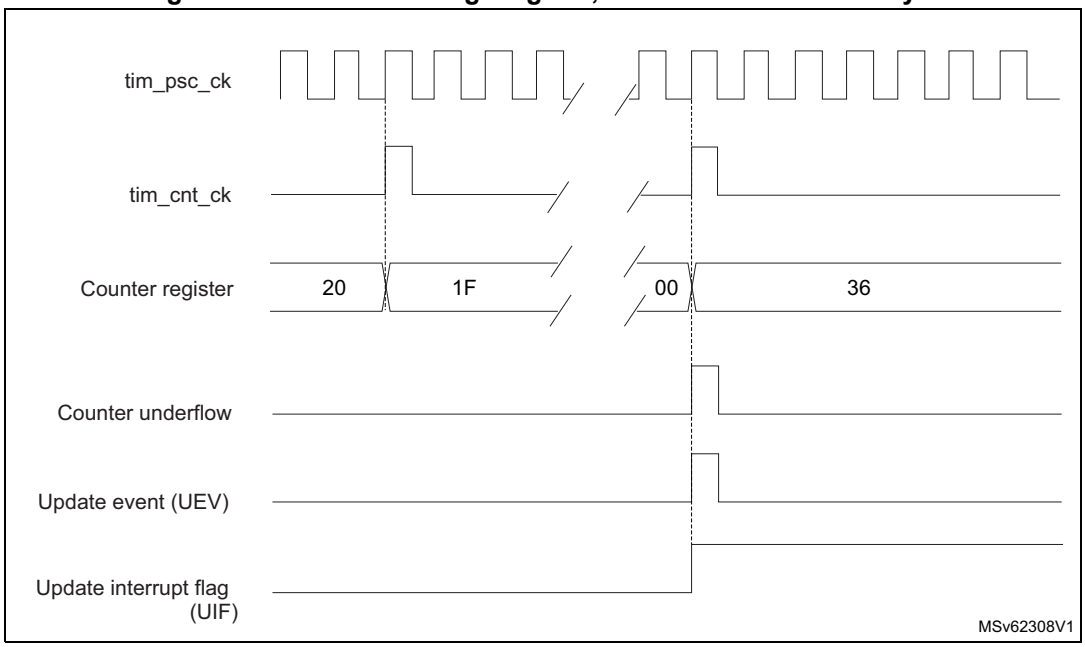
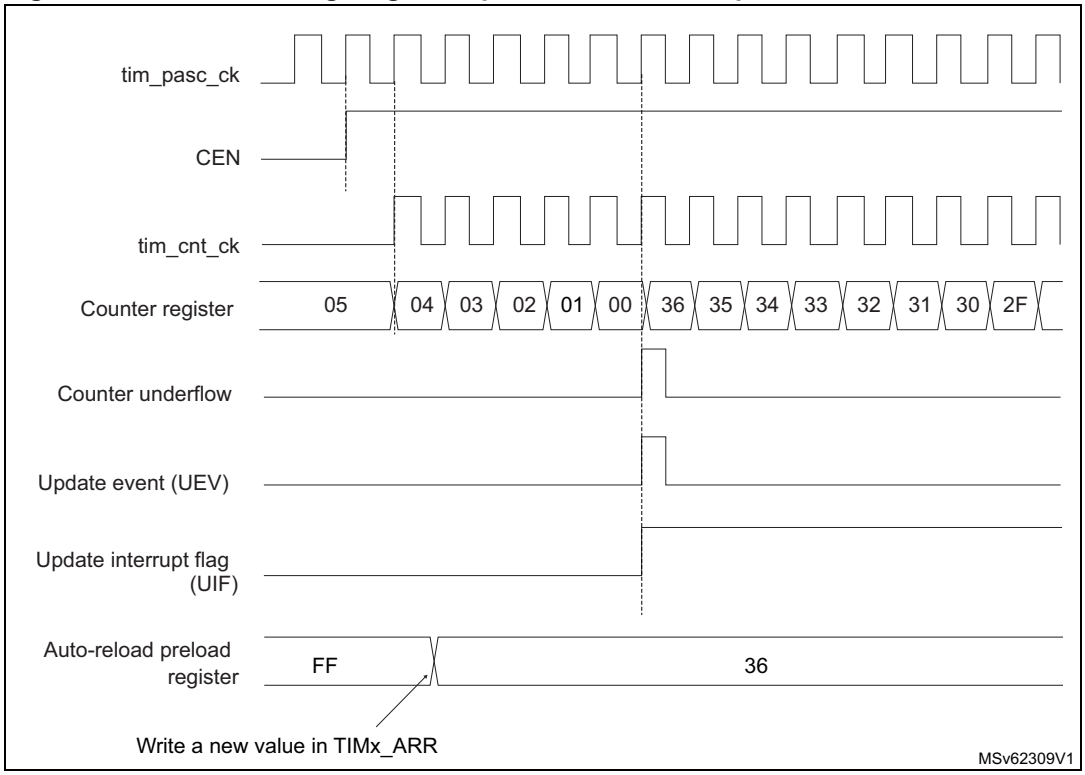


Figure 356. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

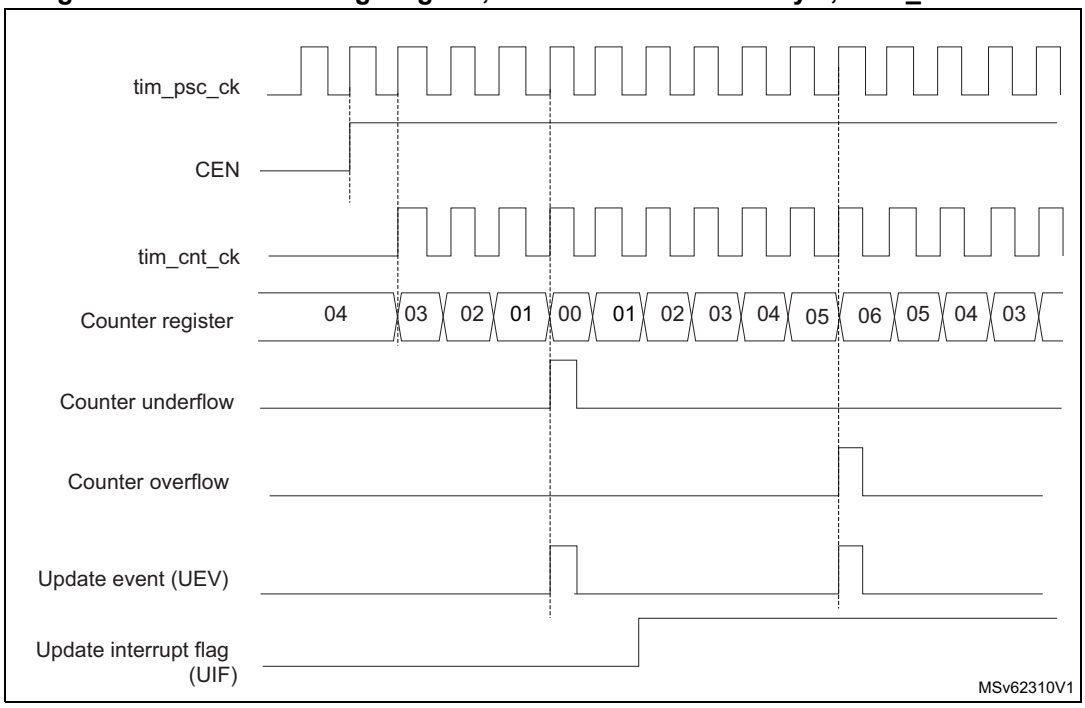
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 357. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 38.6: TIM1/TIM8 registers*).

Figure 358. Counter timing diagram, internal clock divided by 2

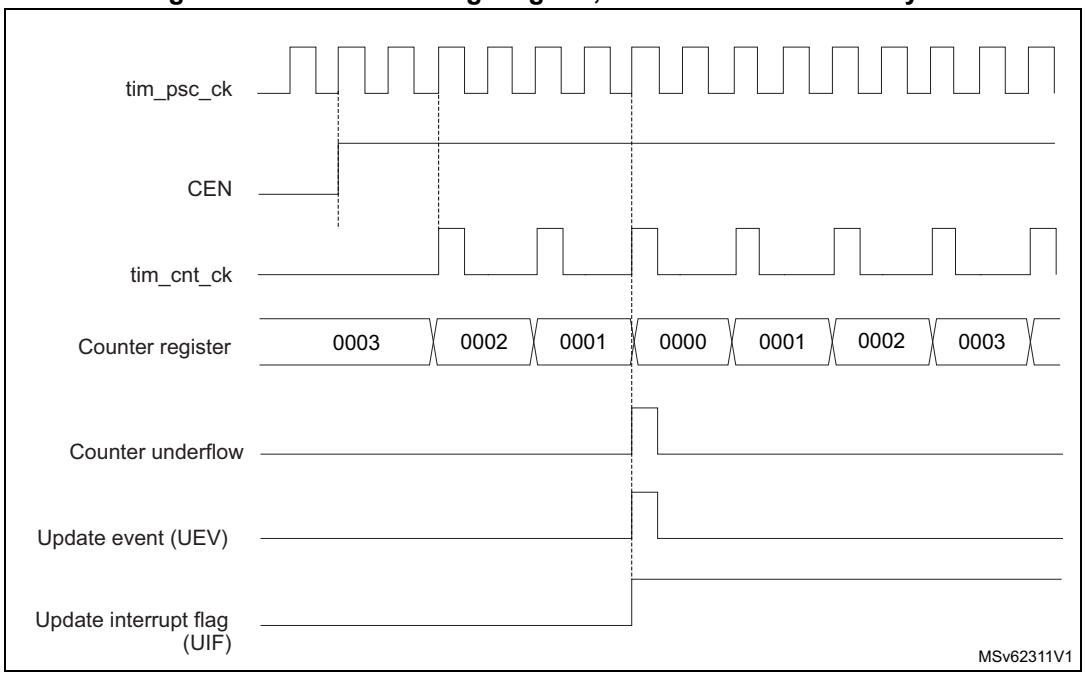


Figure 359. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36

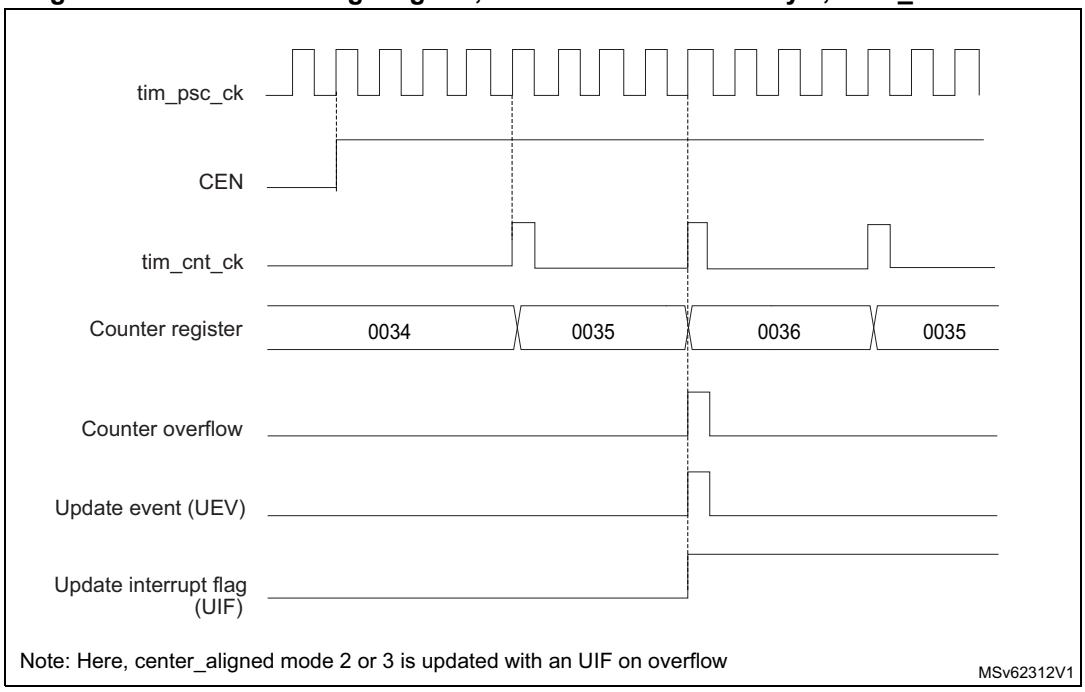


Figure 360. Counter timing diagram, internal clock divided by N

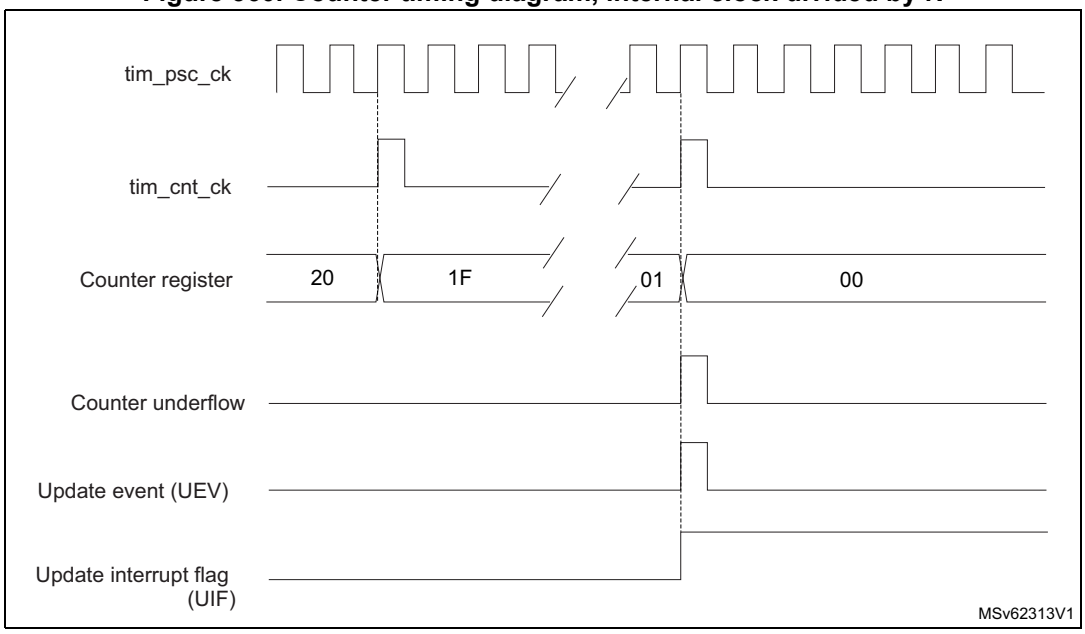


Figure 361. Counter timing diagram, update event with ARPE = 1 (counter underflow)

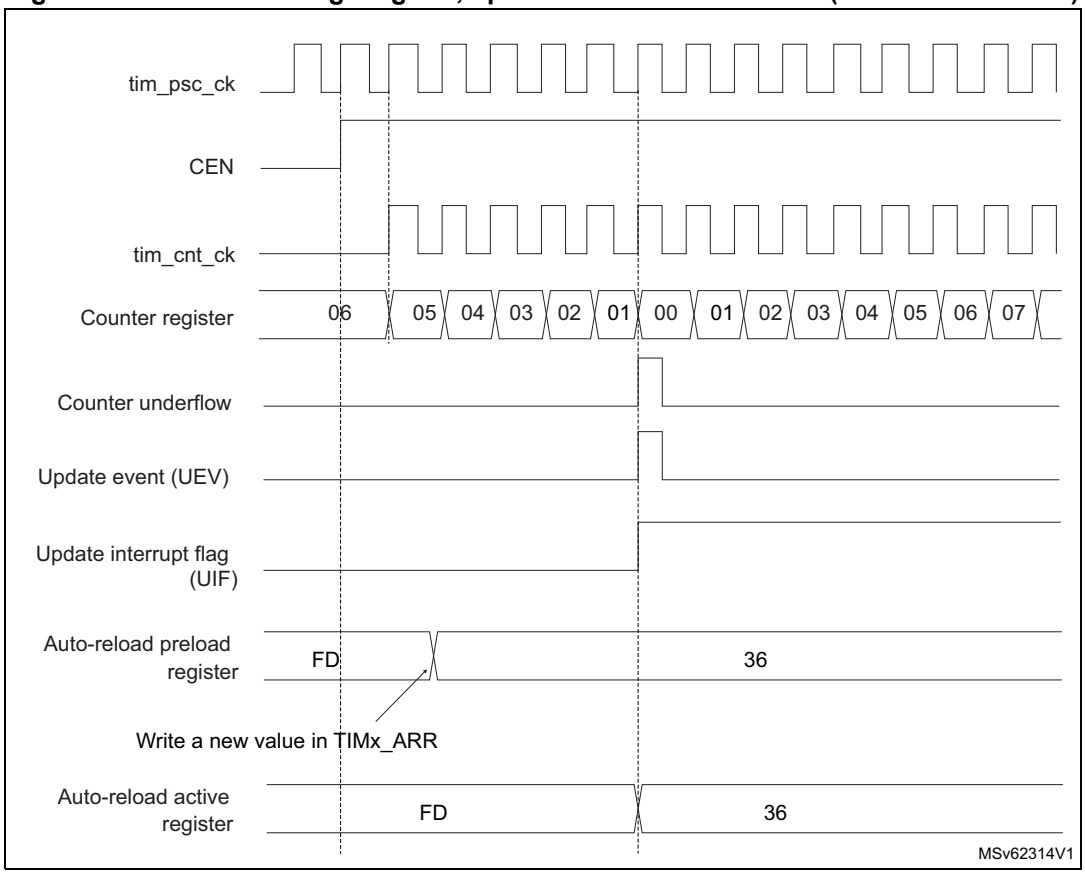
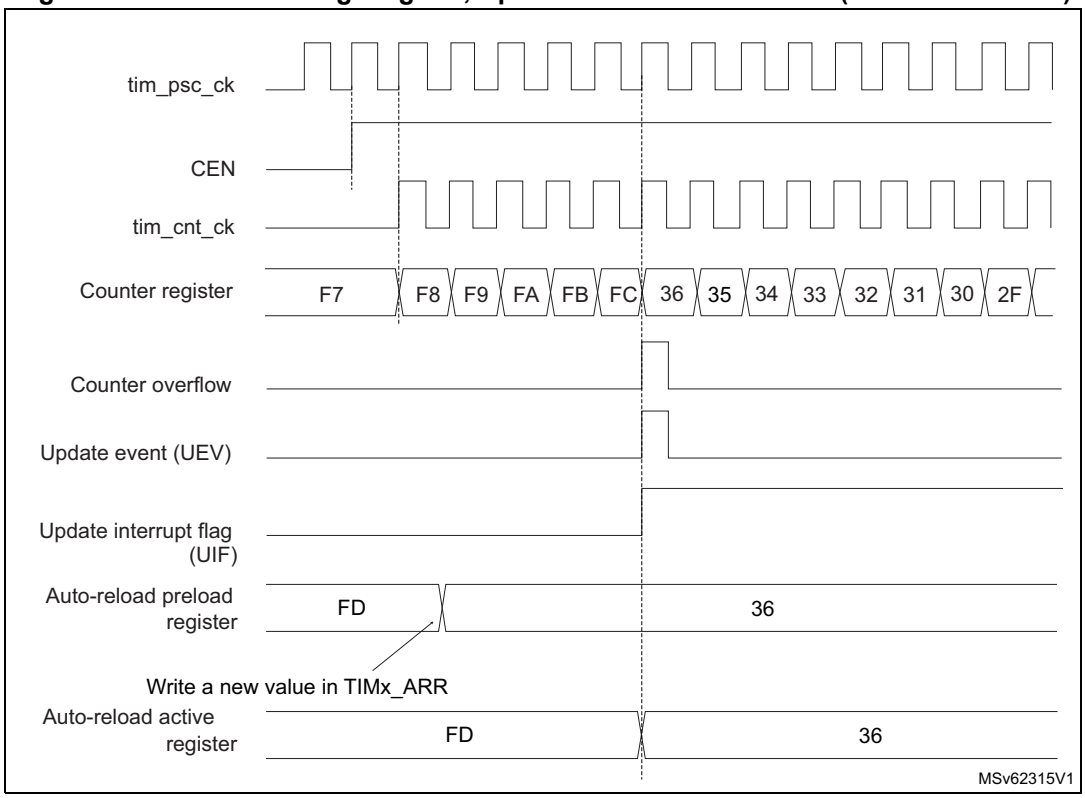


Figure 362. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



38.3.5 Repetition counter

Section 38.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

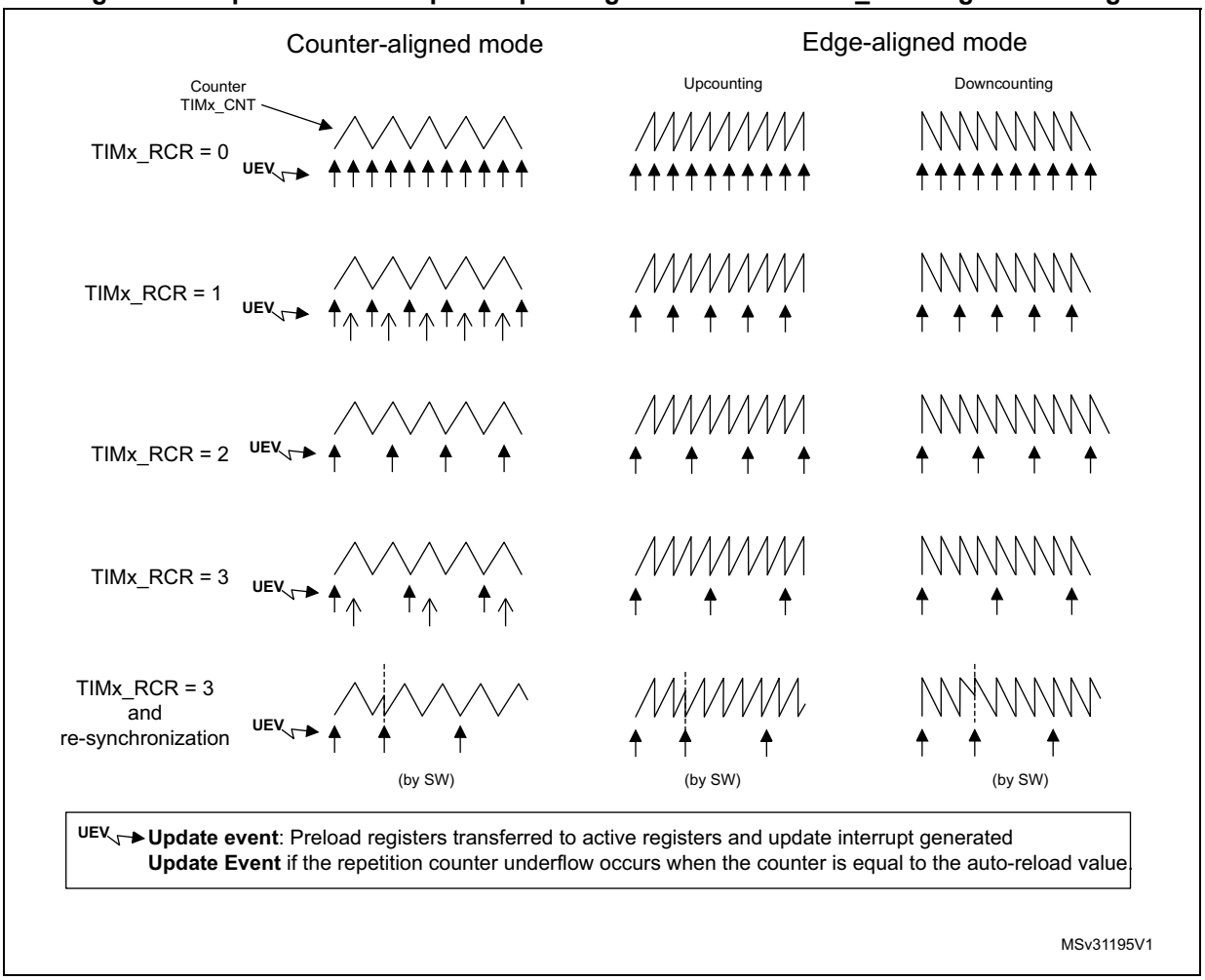
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 363*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 363. Update rate examples depending on mode and TIMx_RCR register settings



38.3.6 External trigger input

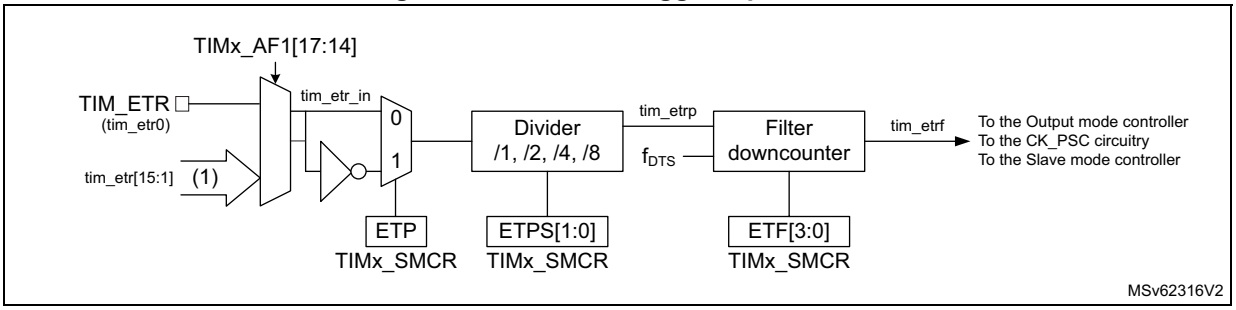
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 38.3.7*)
- trigger for the slave mode (see *Section 38.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 38.3.9*)

Figure 364 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition

the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 364. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for the list of sources connected to the etr_in input in the product.

38.3.7 Clock selection

The counter clock can be provided by the following clock sources:

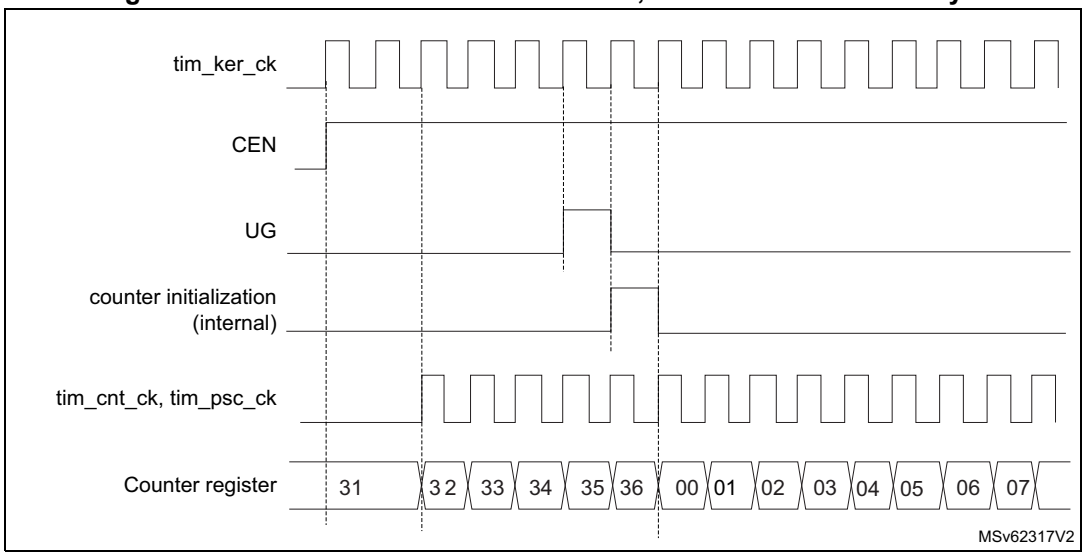
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 365 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

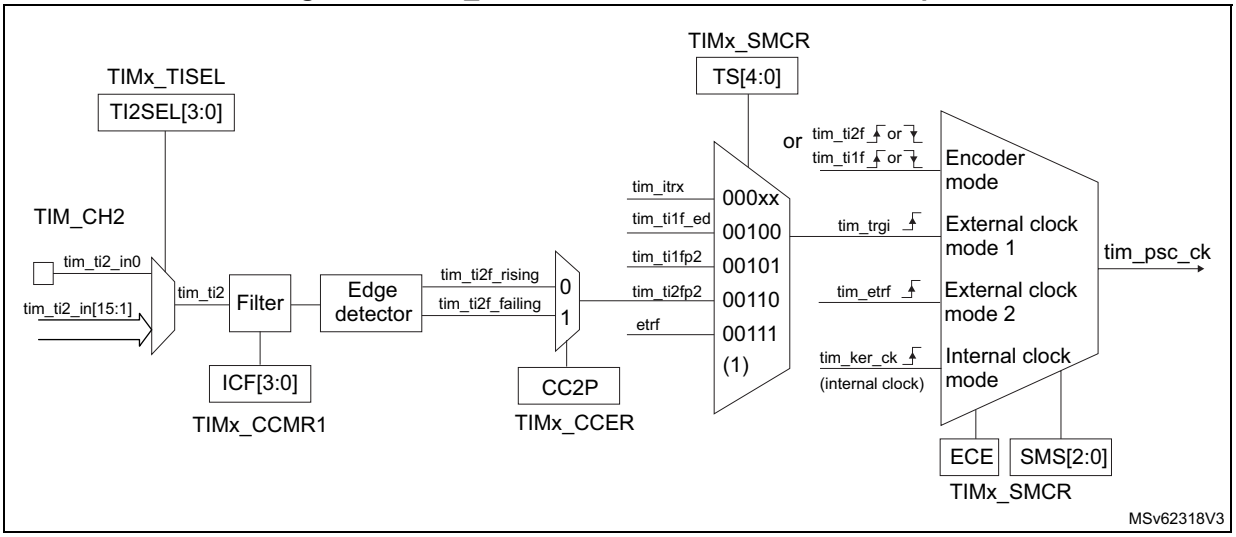
Figure 365. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 366. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

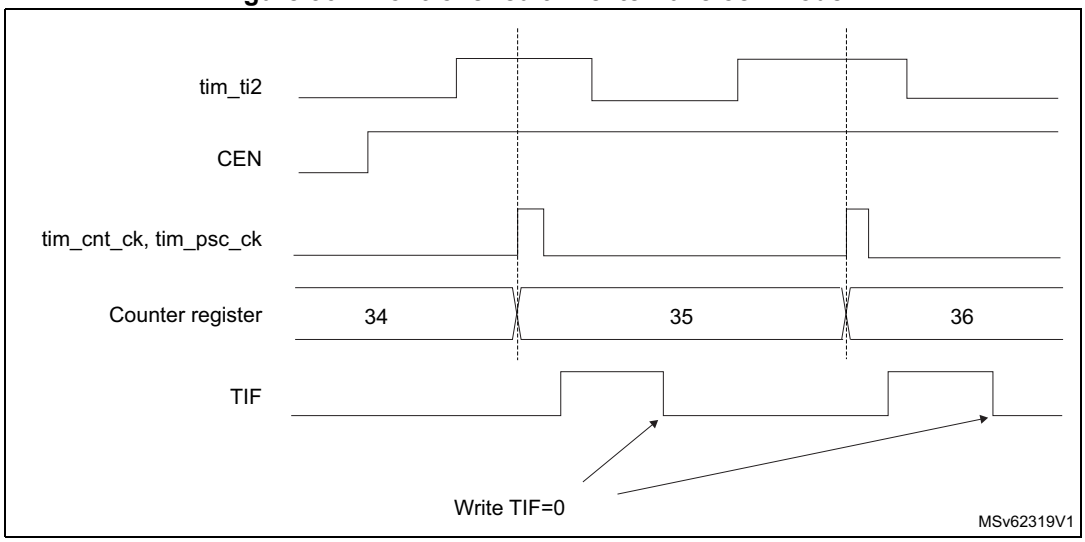
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 367. Control circuit in external clock mode 1



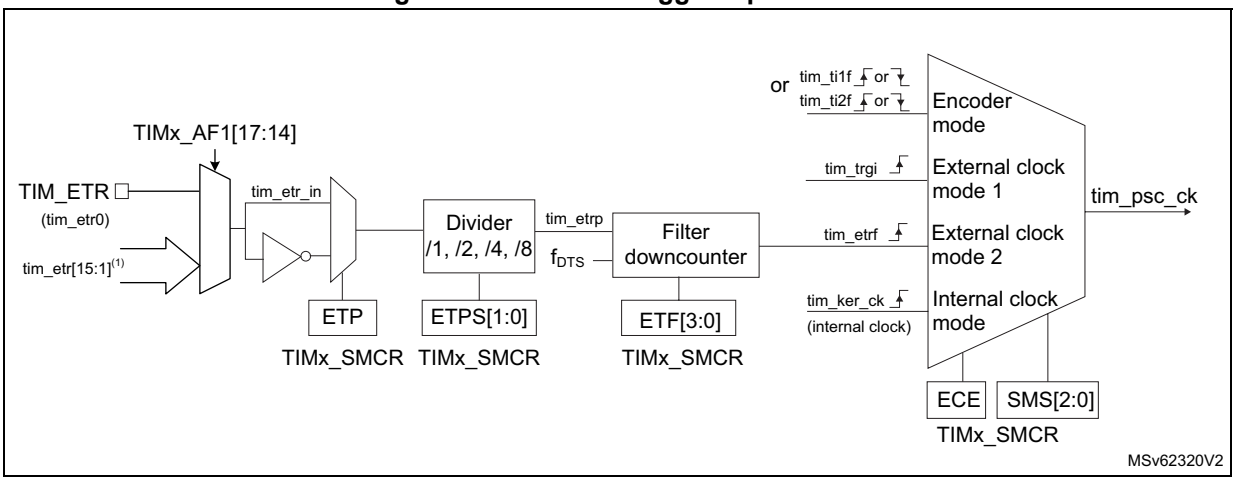
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 368* gives an overview of the external trigger input block.

Figure 368. External trigger input block



1. Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals*.

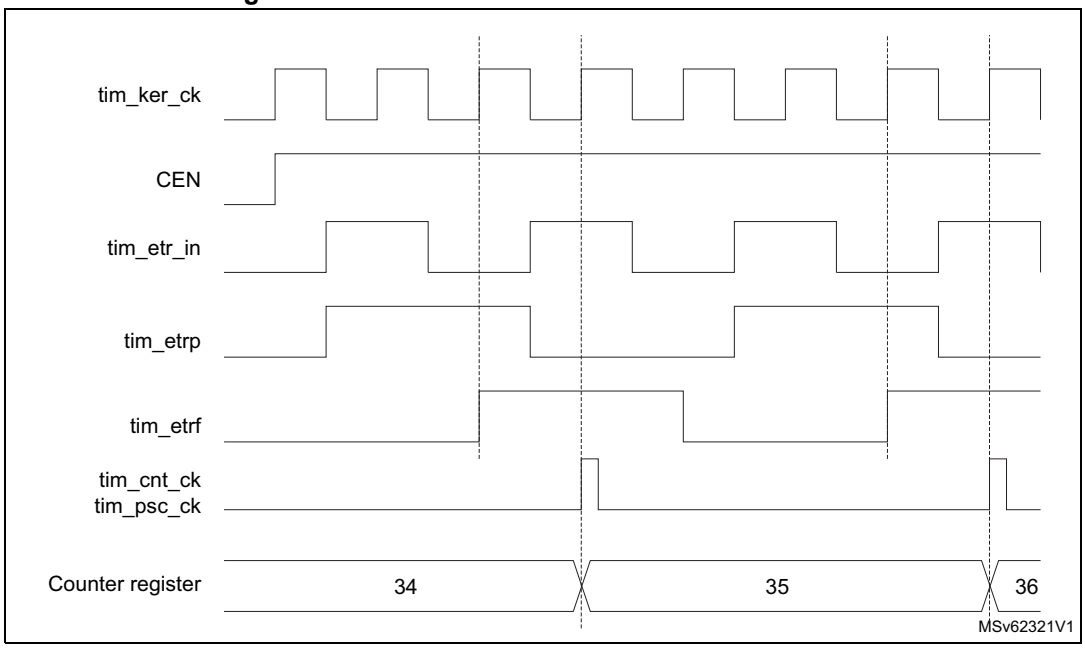
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 369. Control circuit in external clock mode 2



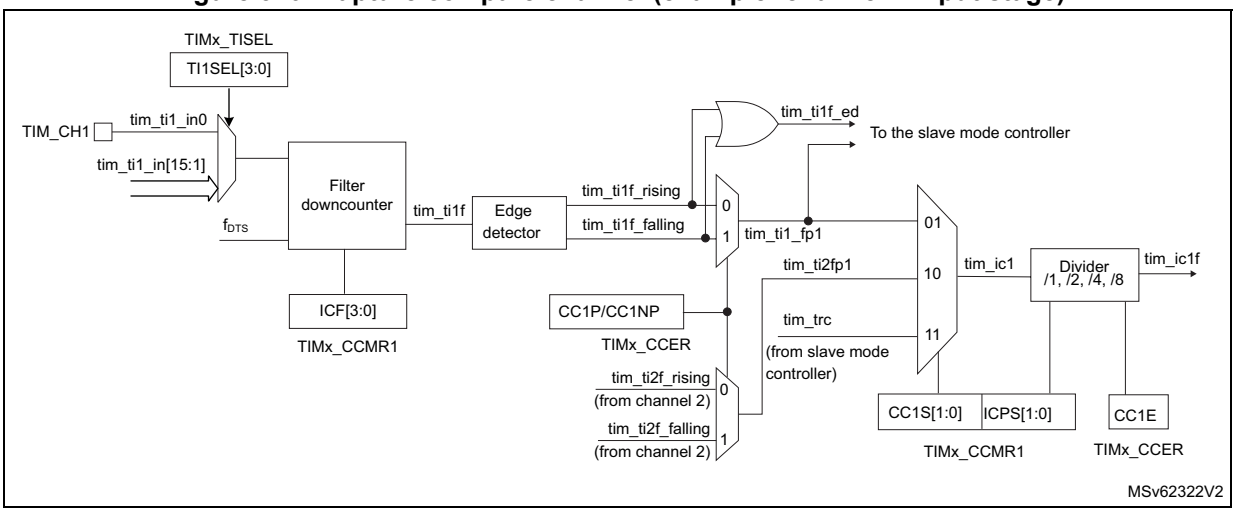
38.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 370 to *Figure 373* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 370. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 371. Capture/compare channel 1 main circuit

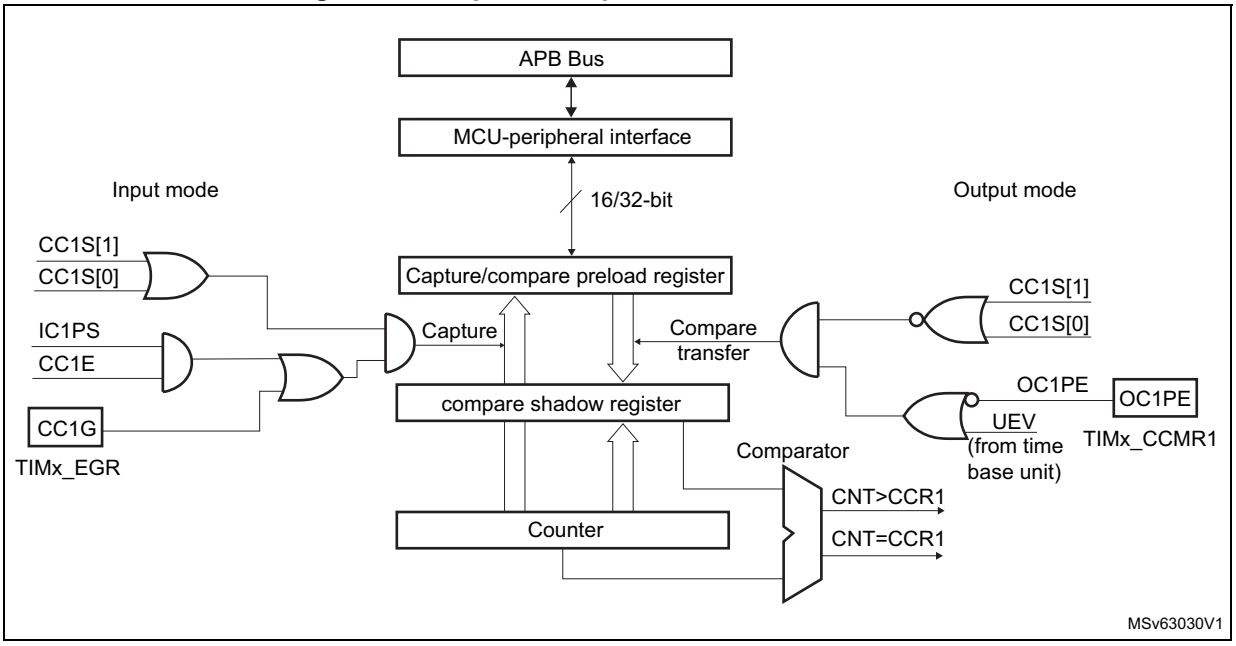
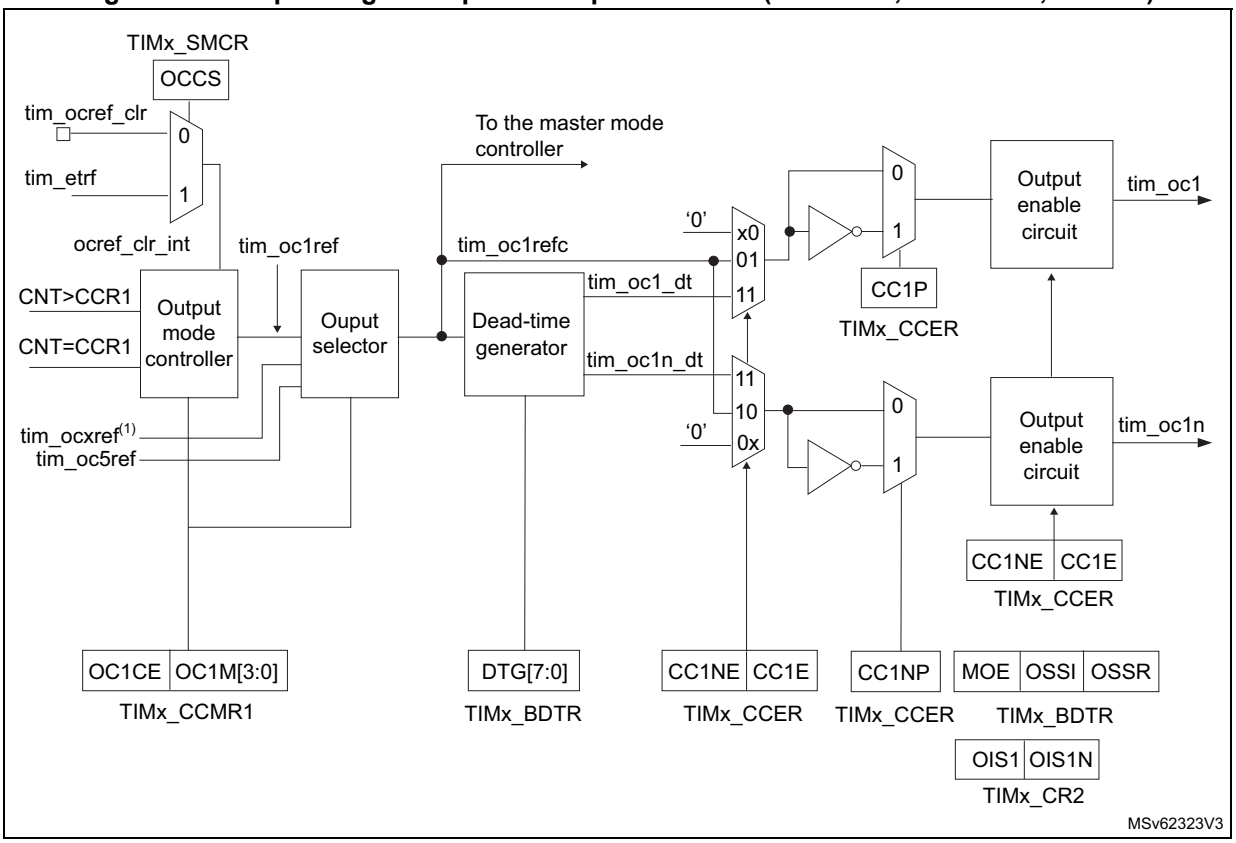
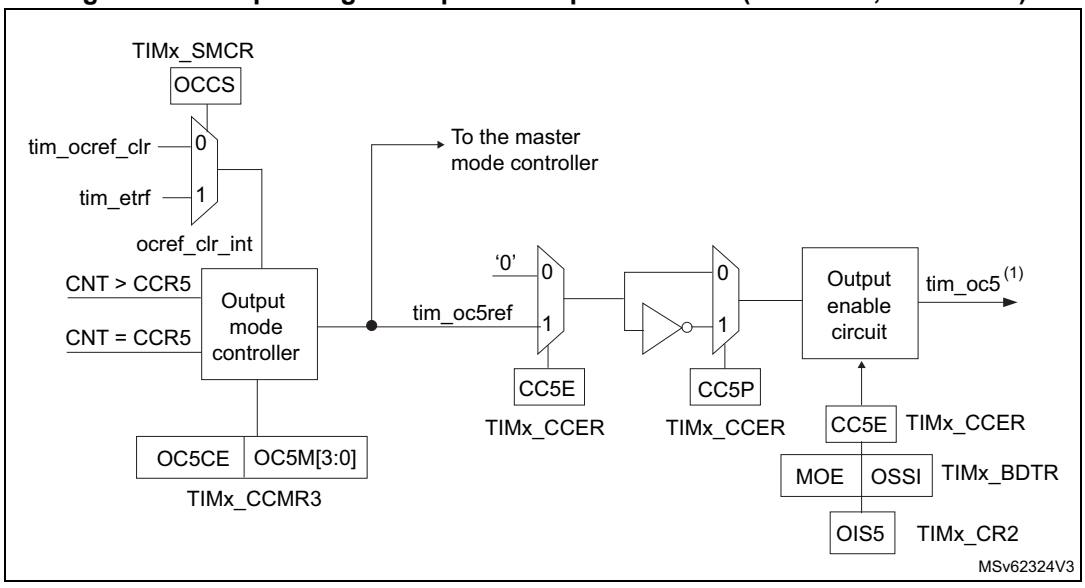


Figure 372. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 373. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

38.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

38.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

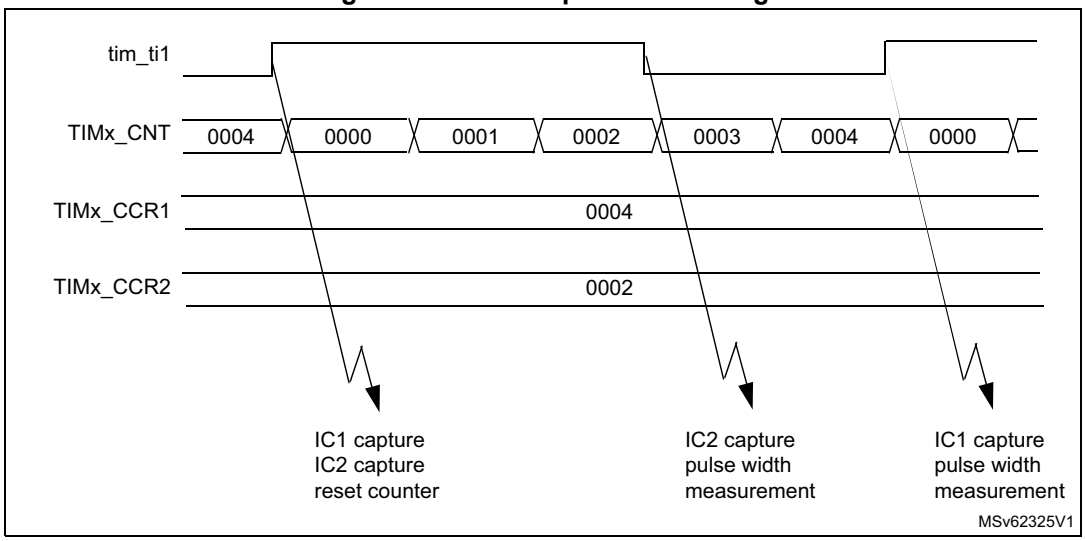
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 374. PWM input mode timing



38.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

38.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 0000), be

set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

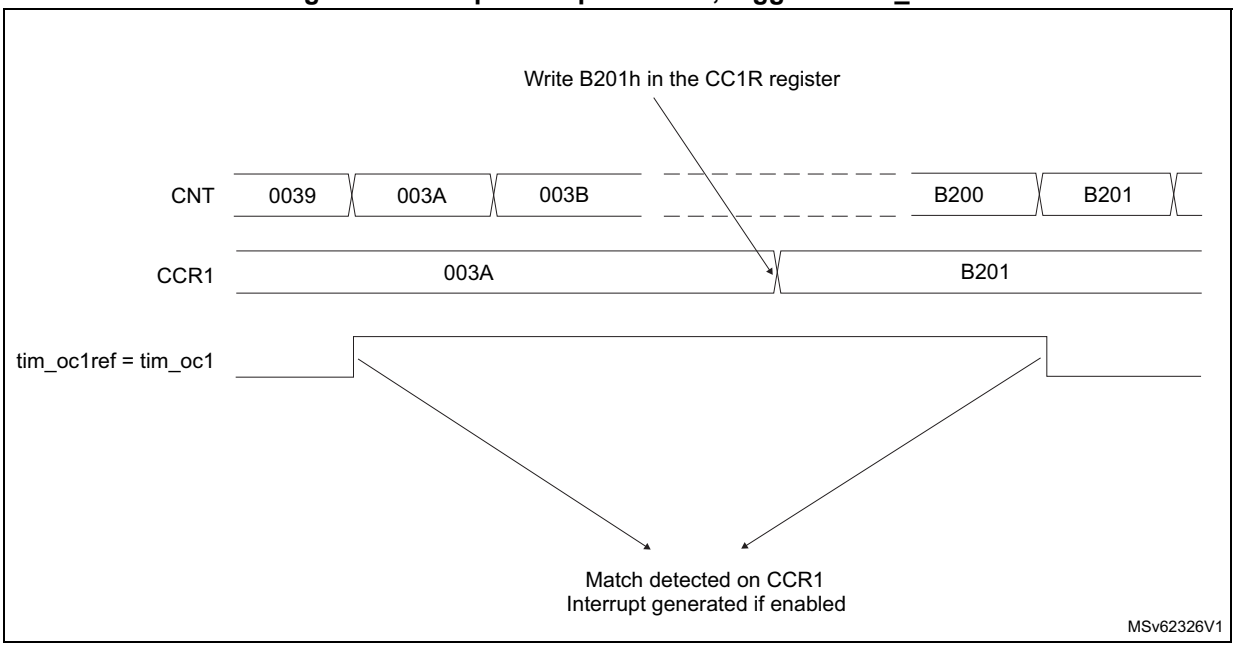
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 375*.

Figure 375. Output compare mode, toggle on tim_oc1



38.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

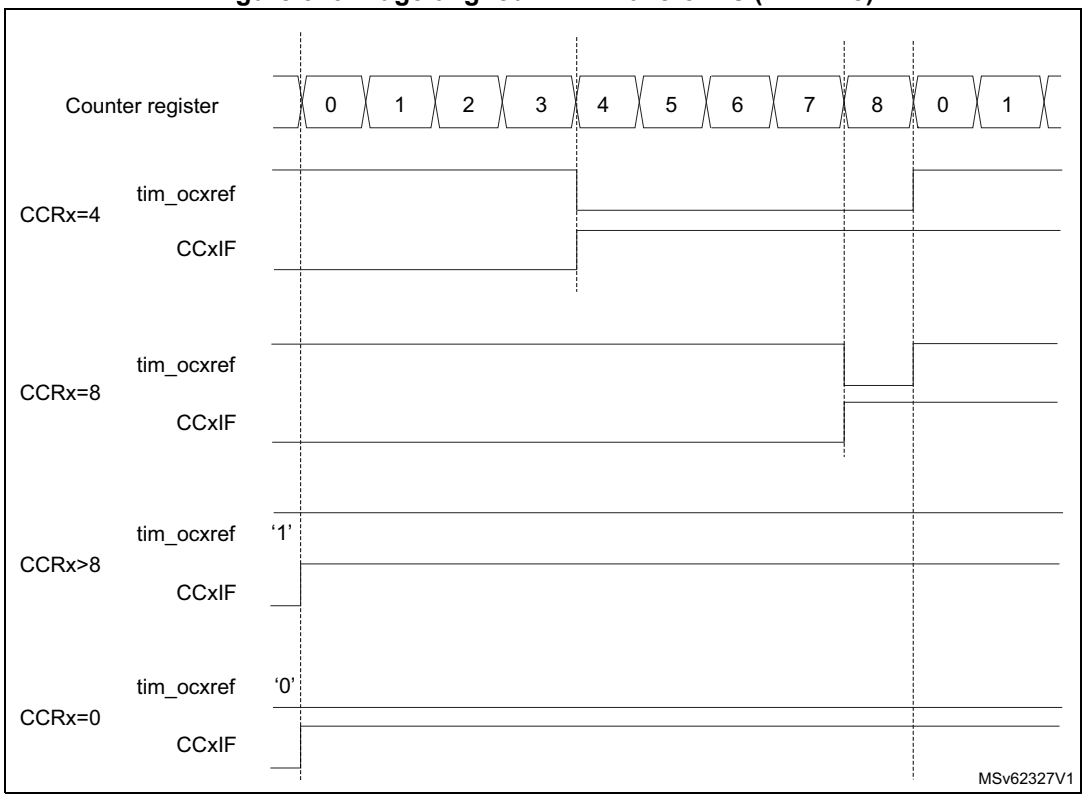
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 376* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 376. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

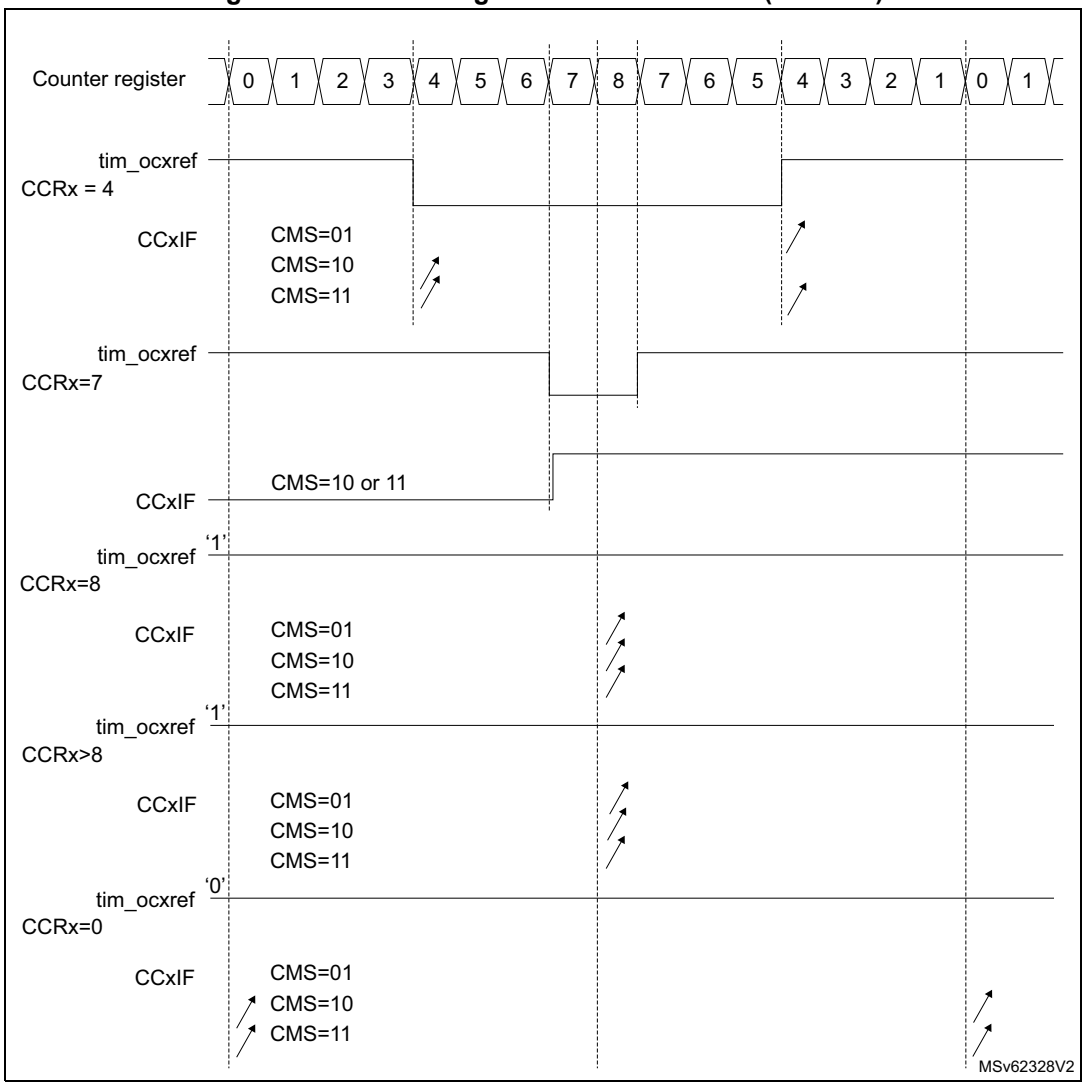
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 377 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 377. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

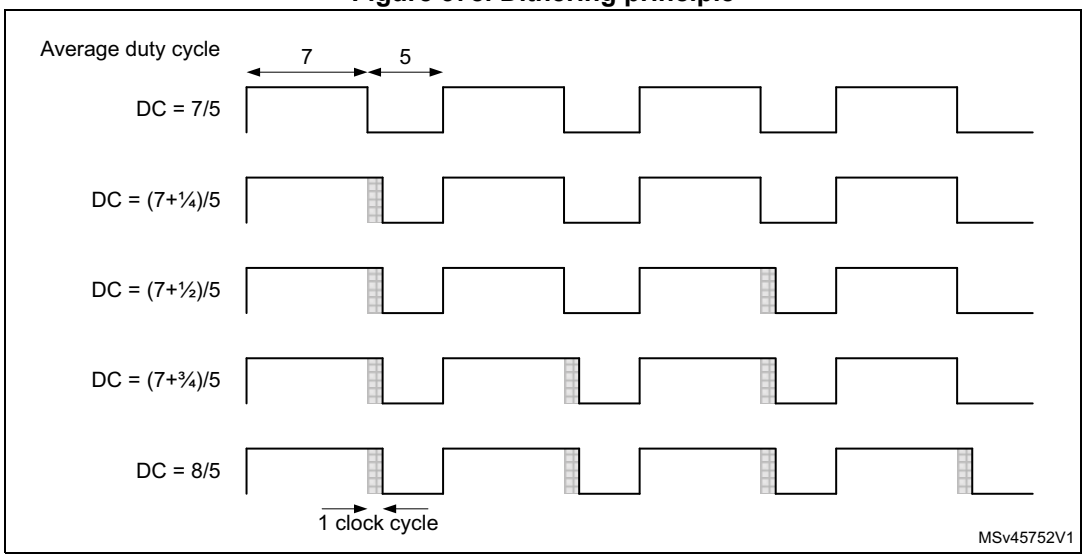
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 378* presents the dithering principle applied to four consecutive PWM cycles.

Figure 378. Dithering principle

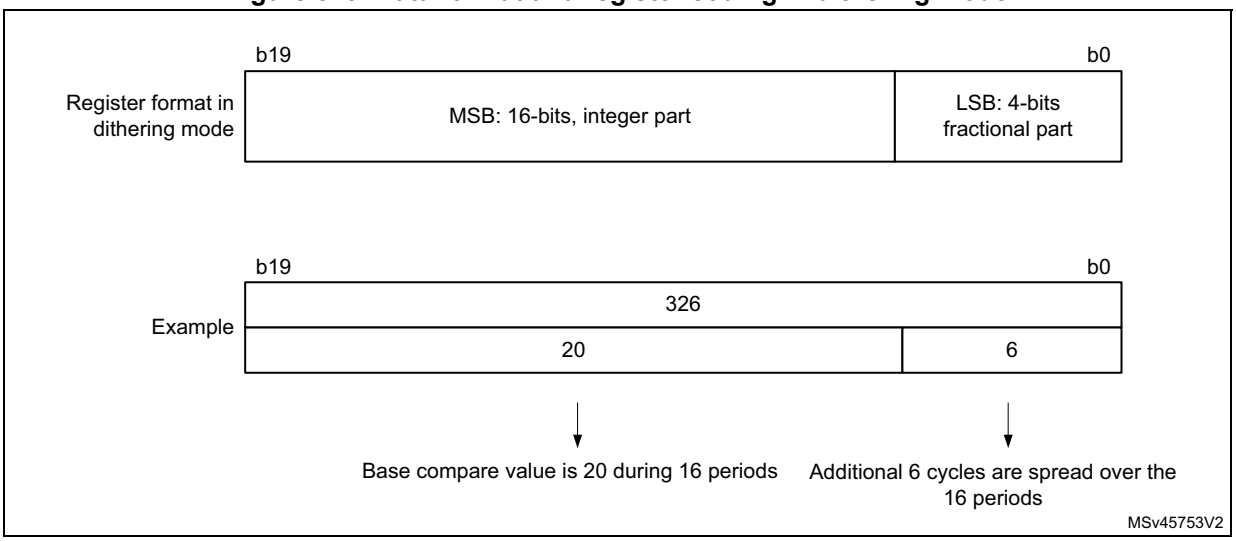


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 379* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset.
2. The DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 379. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

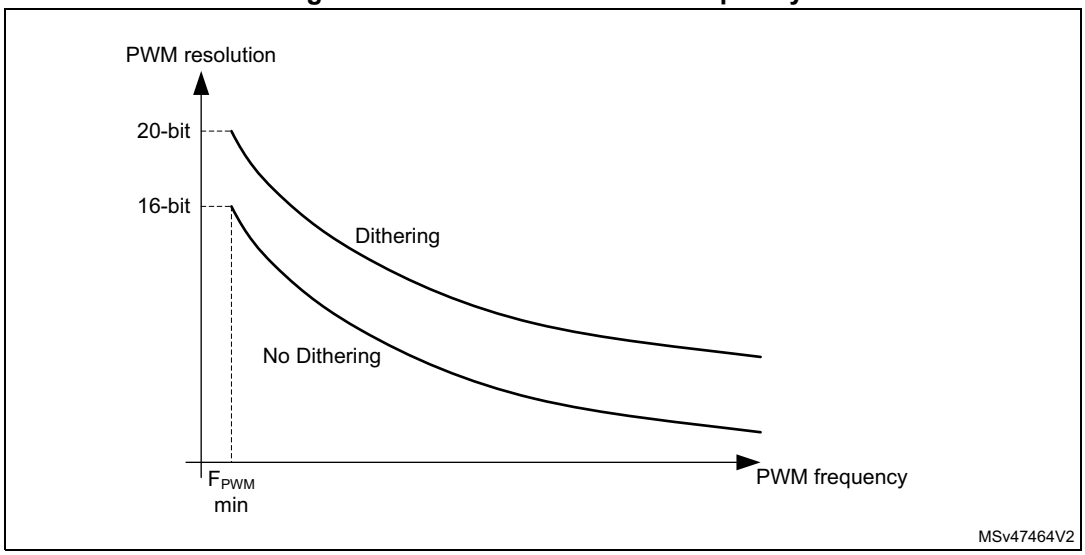
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

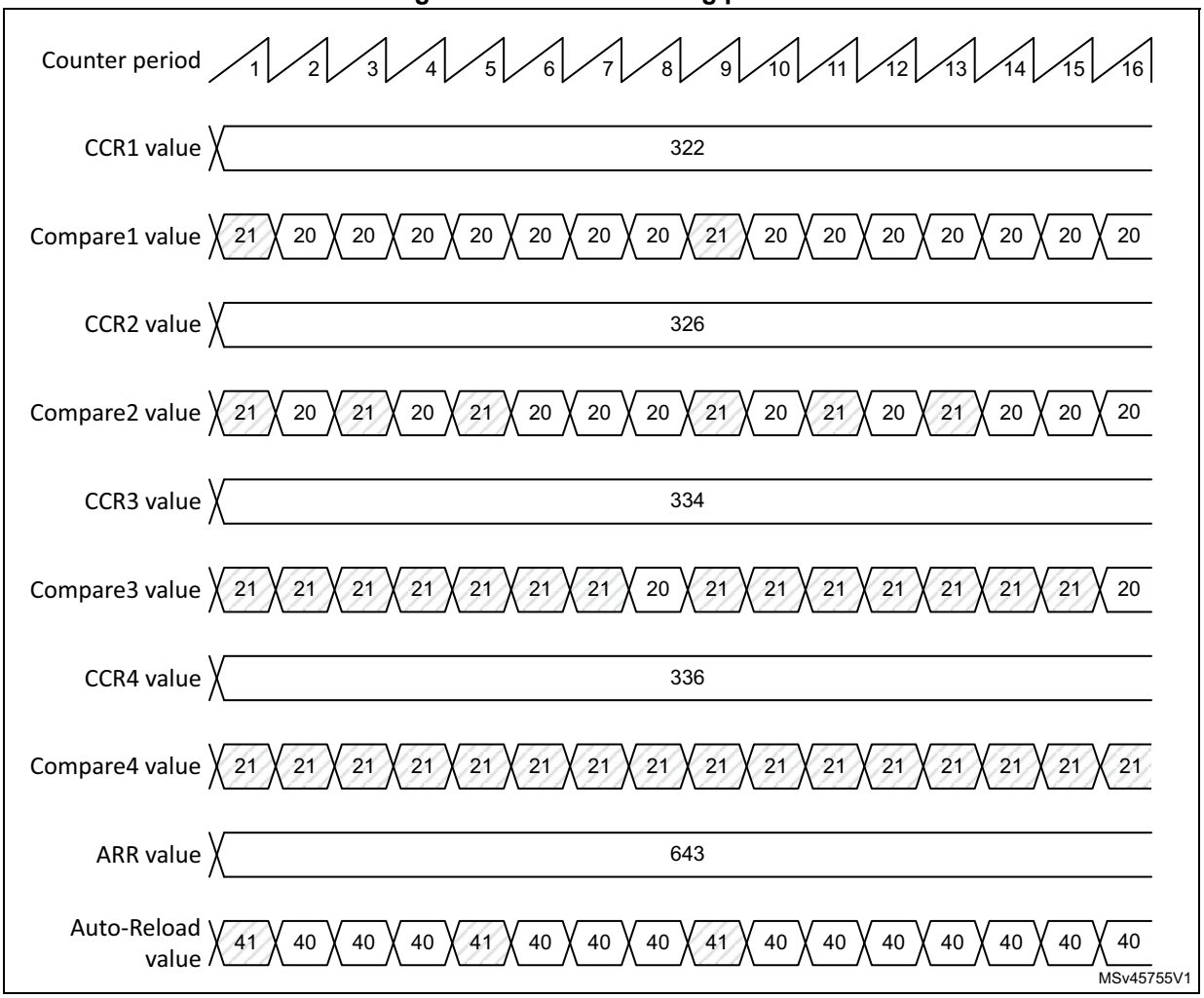
As shown on *Figure 380*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 380. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 381*.

Figure 381. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 403*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 403. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 403. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 382*.

Figure 382. Dithering effect on duty cycle in center-aligned PWM mode

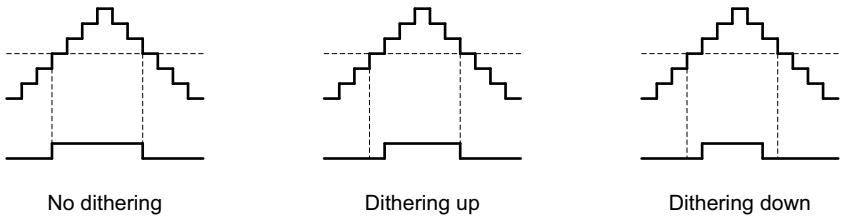


Table 404 shows how the dithering pattern is added in center-aligned PWM mode.

Table 404. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 404. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

38.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

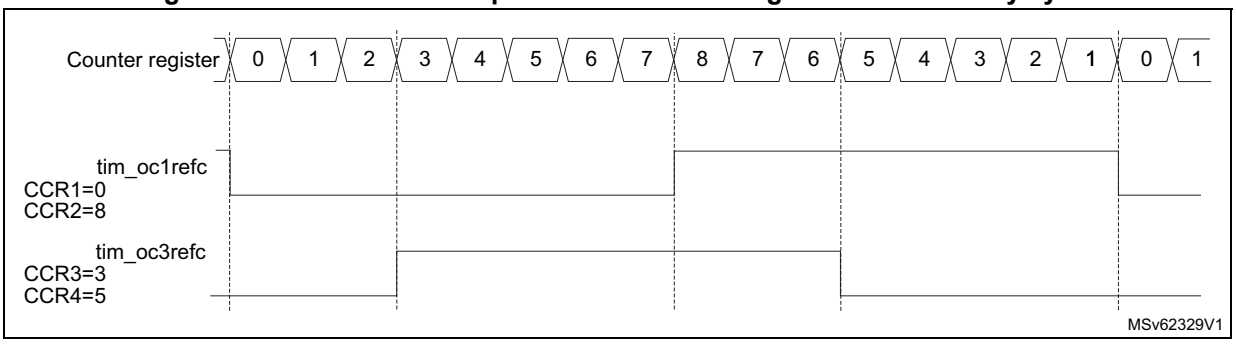
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 383 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 383. Generation of 2 phase-shifted PWM signals with 50% duty cycle



38.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

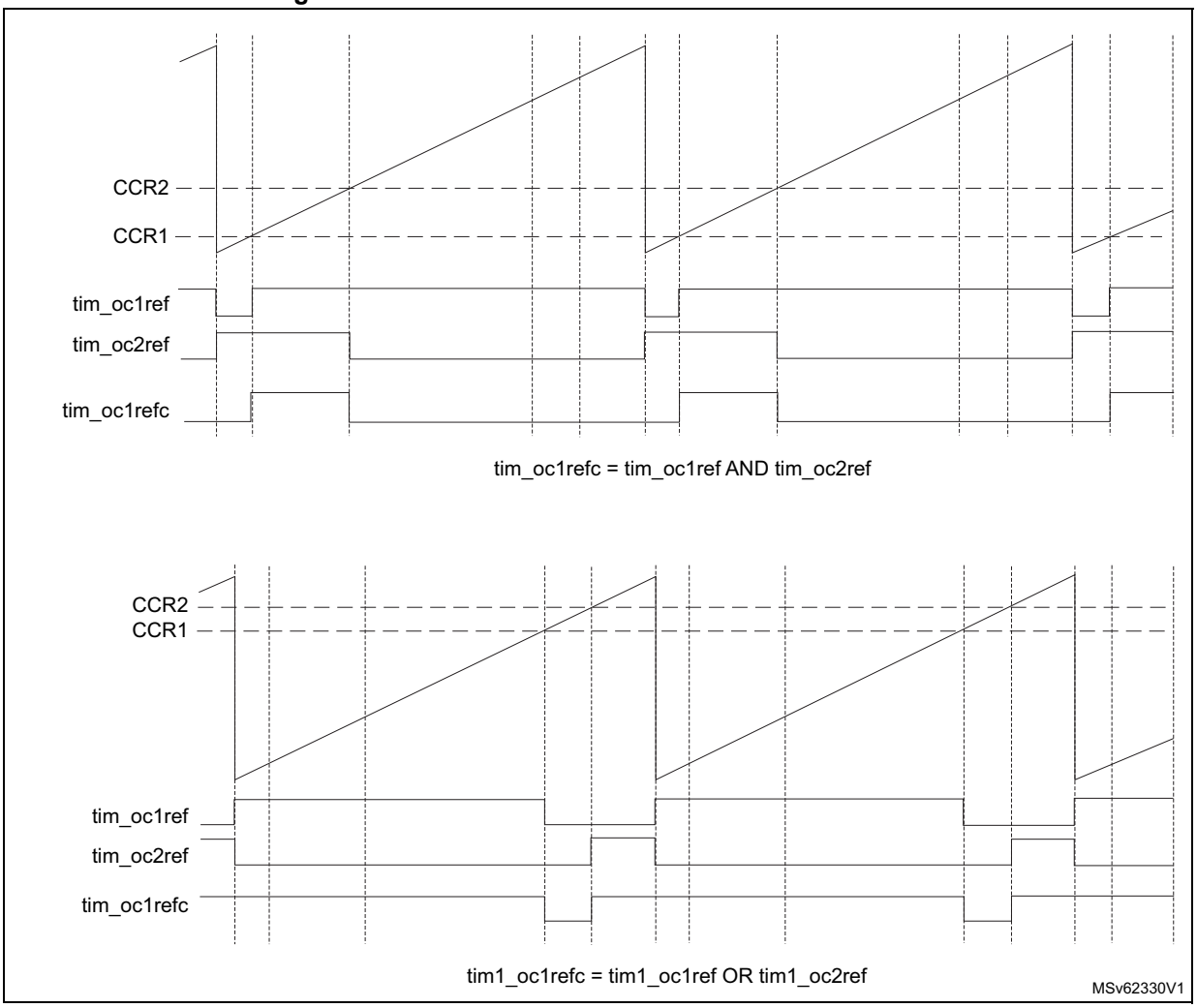
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 384 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 384. Combined PWM mode on channel 1 and 3



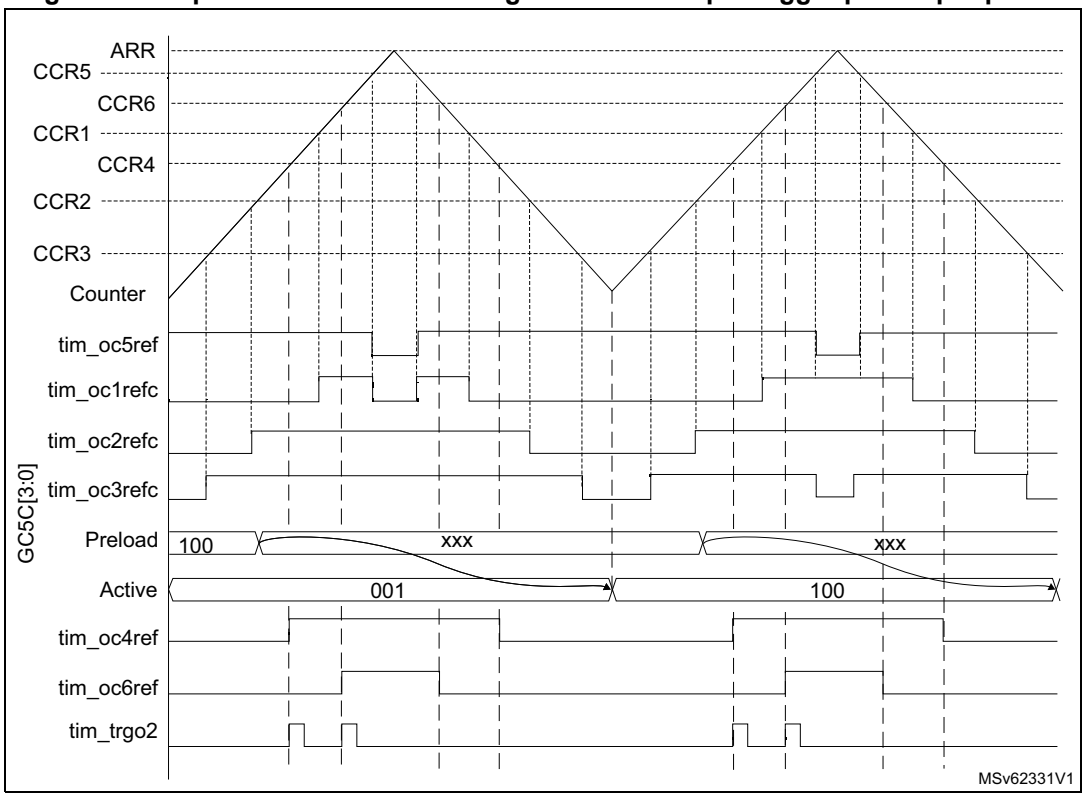
38.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5.
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5.
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5.

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 385. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 38.3.31: ADC triggers* for more details.

38.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 412: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

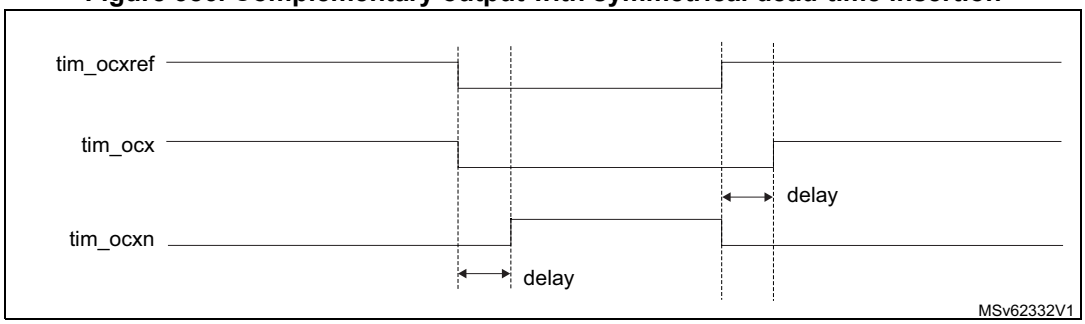
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 386. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 387*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 387. Asymmetrical deadtime

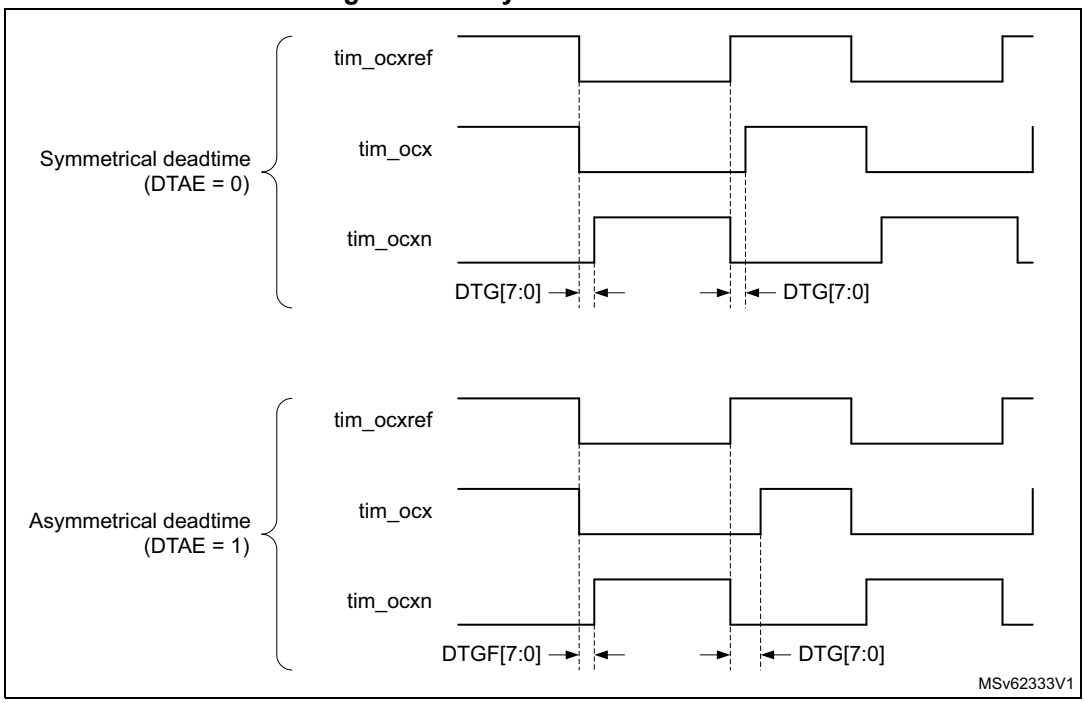


Figure 388. Dead-time waveforms with delay greater than the negative pulse

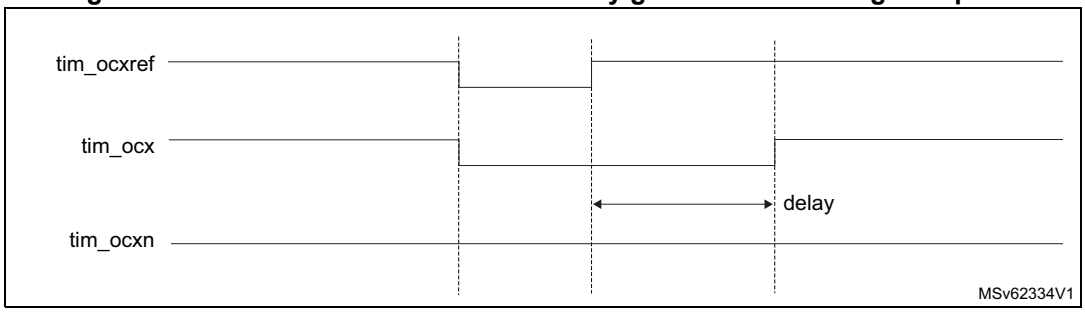
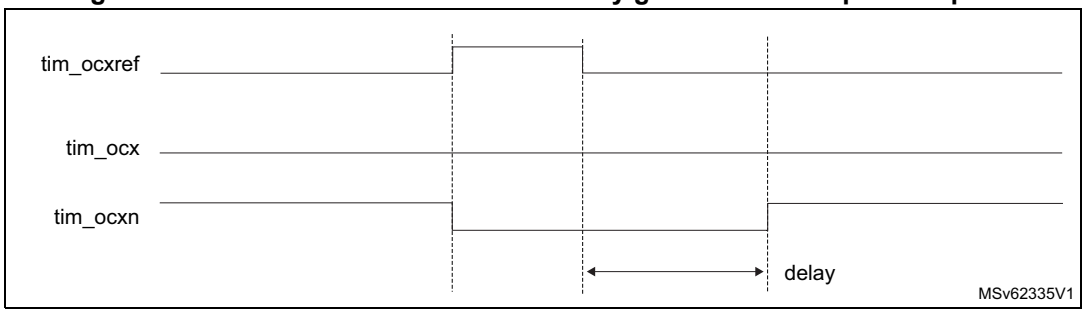


Figure 389. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 38.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

38.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 412: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous

and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

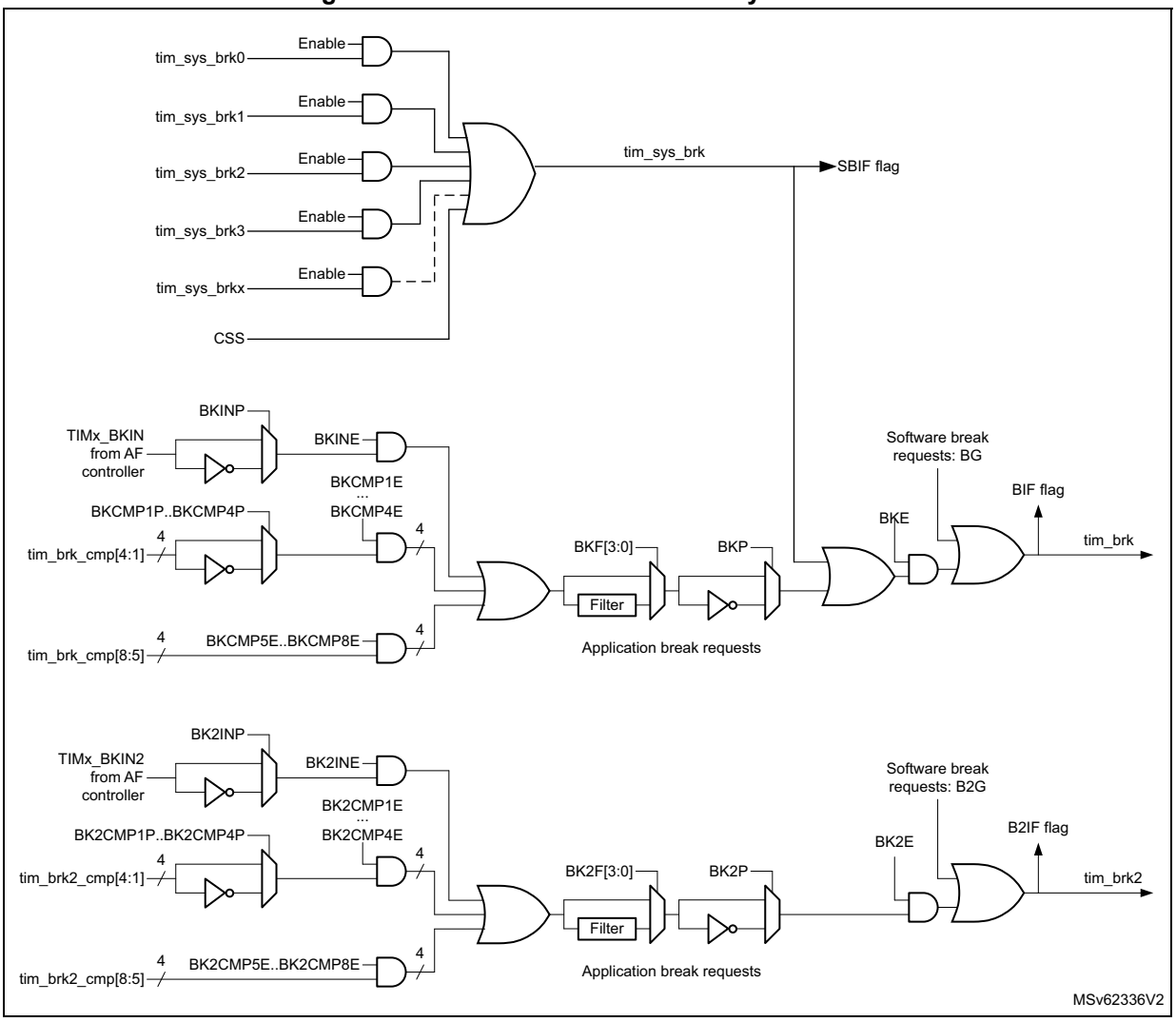
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 390* below.

Figure 390. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF, and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

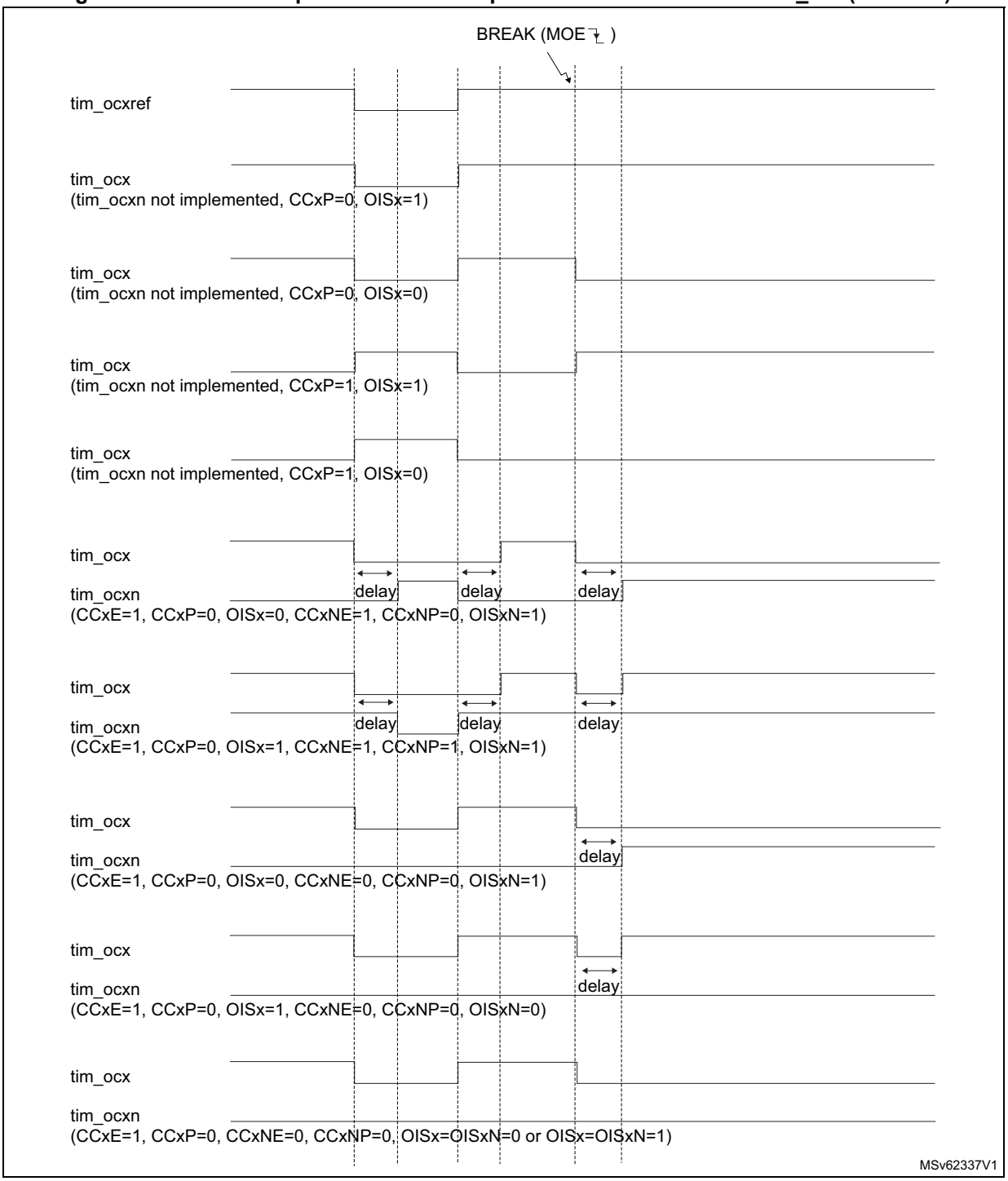
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 38.6.20*. The LOCK bits can be written only once after an MCU reset.

Figure 391 shows an example of behavior of the outputs in response to a break.

Figure 391. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 405*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 405. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 392 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 392. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

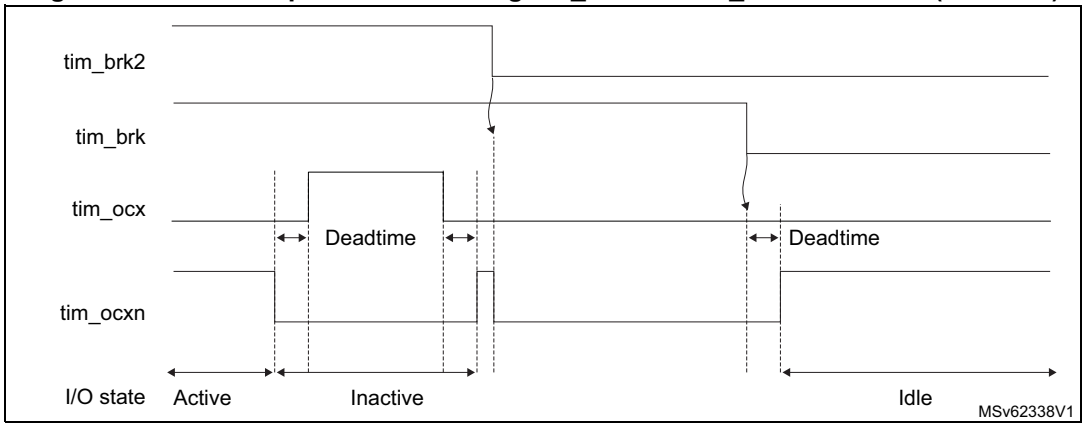
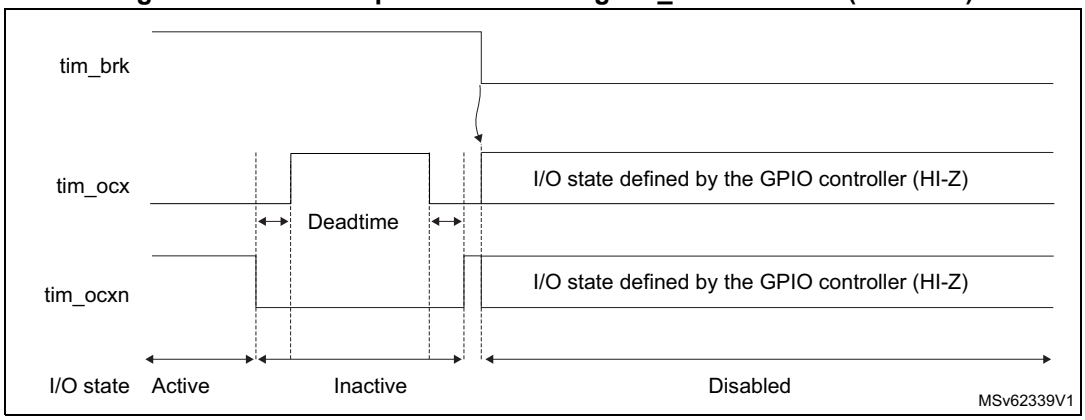


Figure 393. PWM output state following tim_brk assertion (OSSI = 0)



38.3.19 Bidirectional break inputs

The TIM1/TIM8 feature bidirectional break I/Os, as represented on *Figure 394*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 406*).

Table 406. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

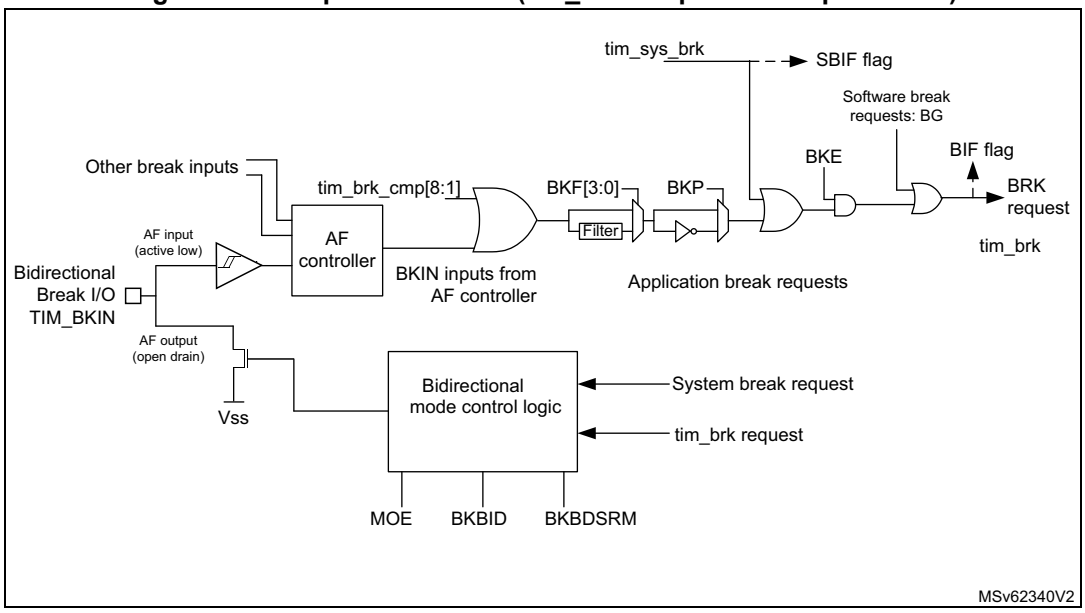
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 394. Output redirection (tim_brk2 request not represented)



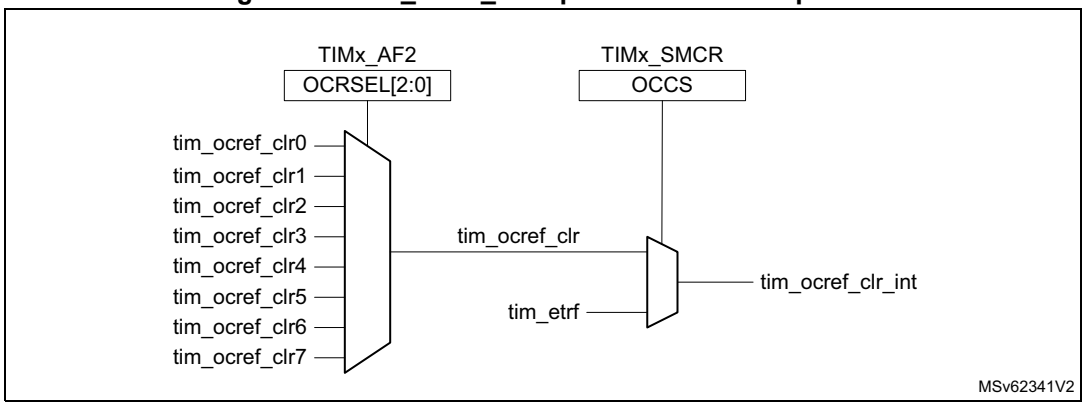
38.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM

cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode. tim_ocref_clr_int input can be selected between the tim_ocref_clr input and tim_etrf (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register.

The tim_ocref_clr input can be selected among several inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown on the *Figure 395* below. Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for a list of sources available in the product.

Figure 395. tim_ocref_clr input selection multiplexer

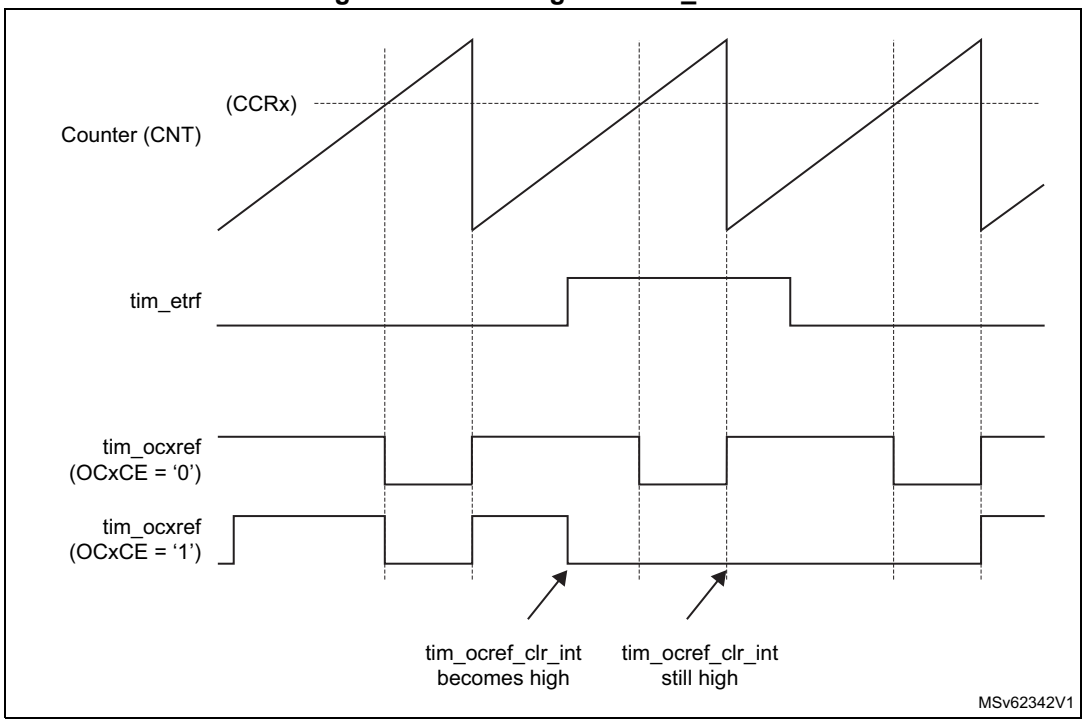


When tim_etrf is chosen, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 396 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 396. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

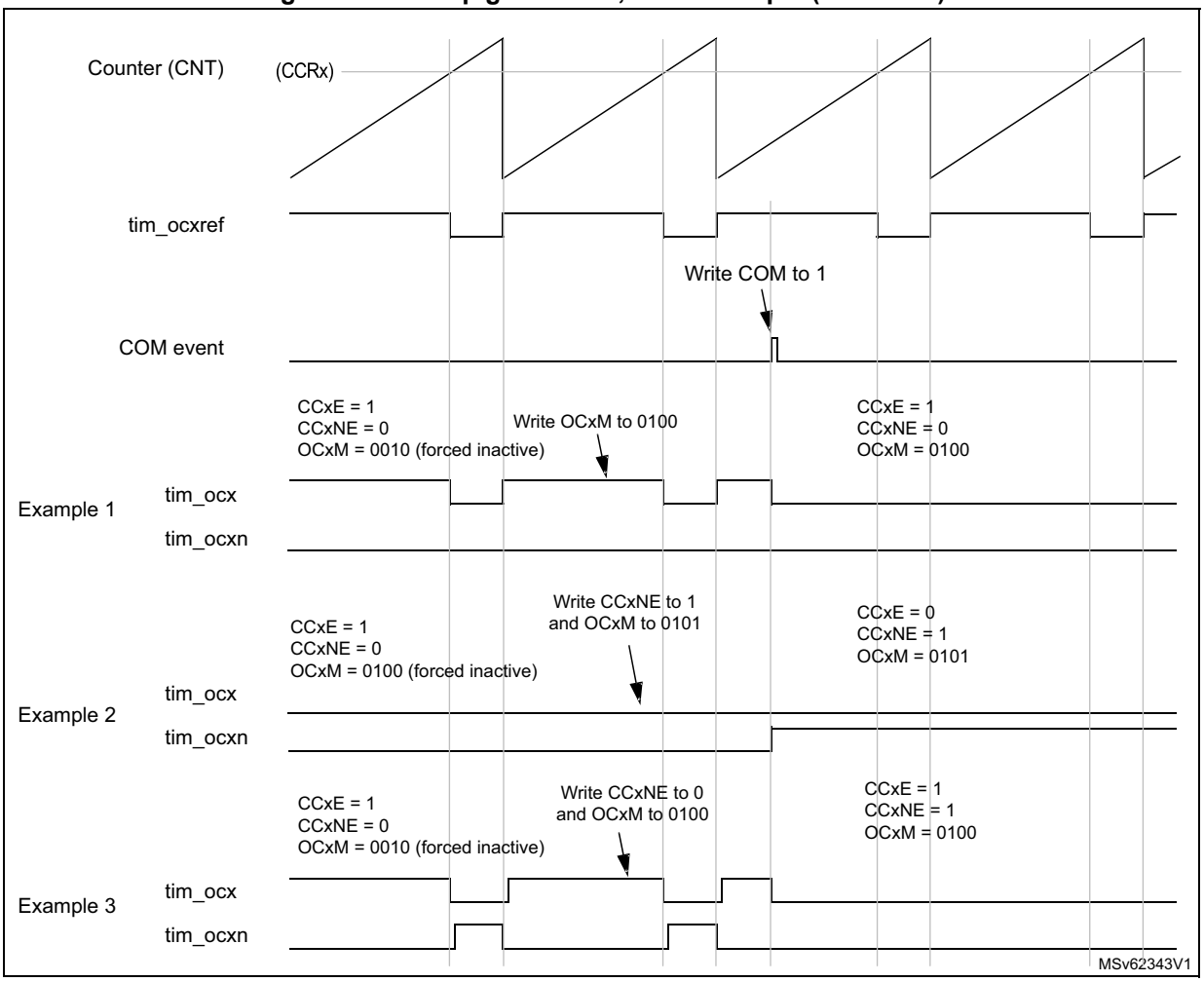
38.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 397 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 397. 6-step generation, COM example (OSSR = 1)



38.3.22 One-pulse mode

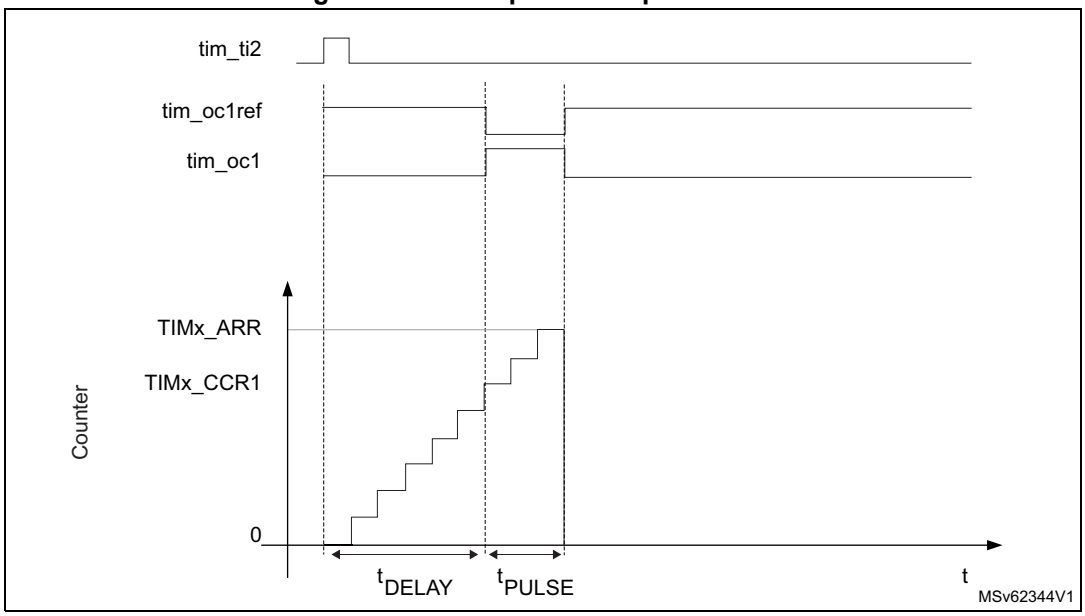
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 398. Example of one pulse mode.



In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 111 in TIMx_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

38.3.23 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in *Section 38.3.22*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

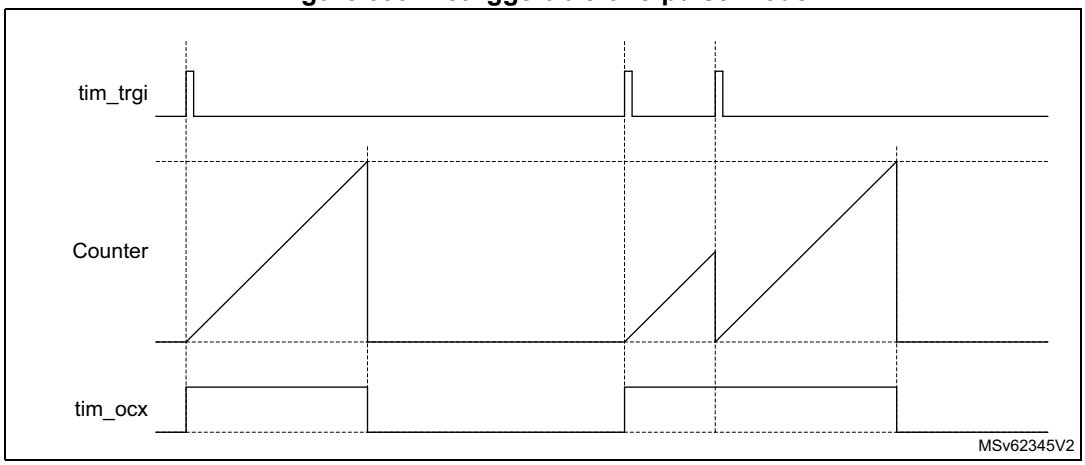
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 399. Retriggerable one-pulse mode



38.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 400*.

Figure 400. Pulse generator circuitry

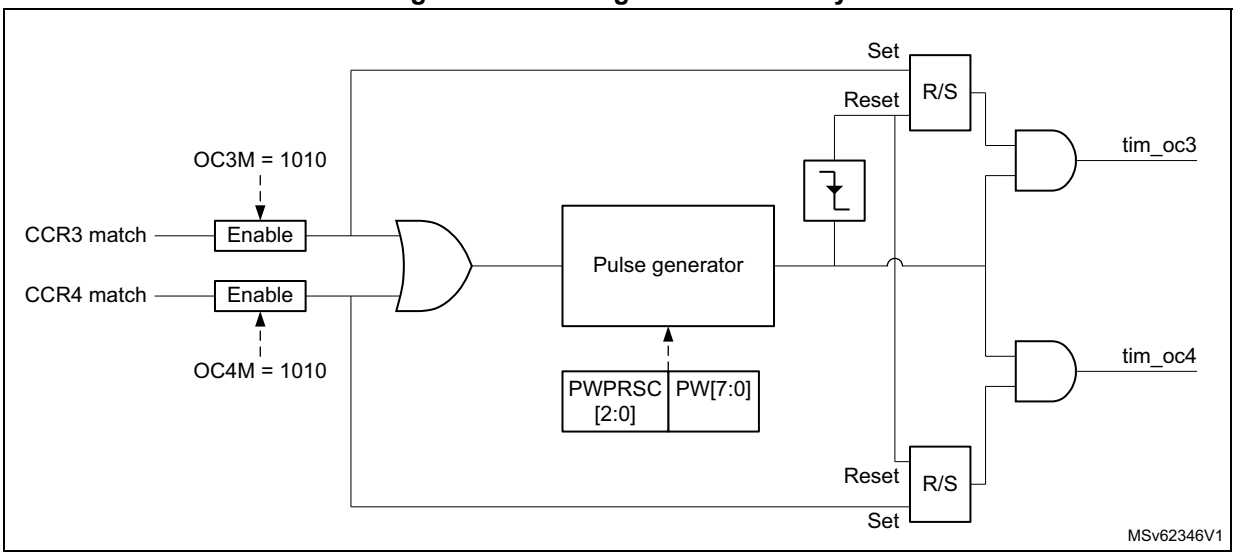
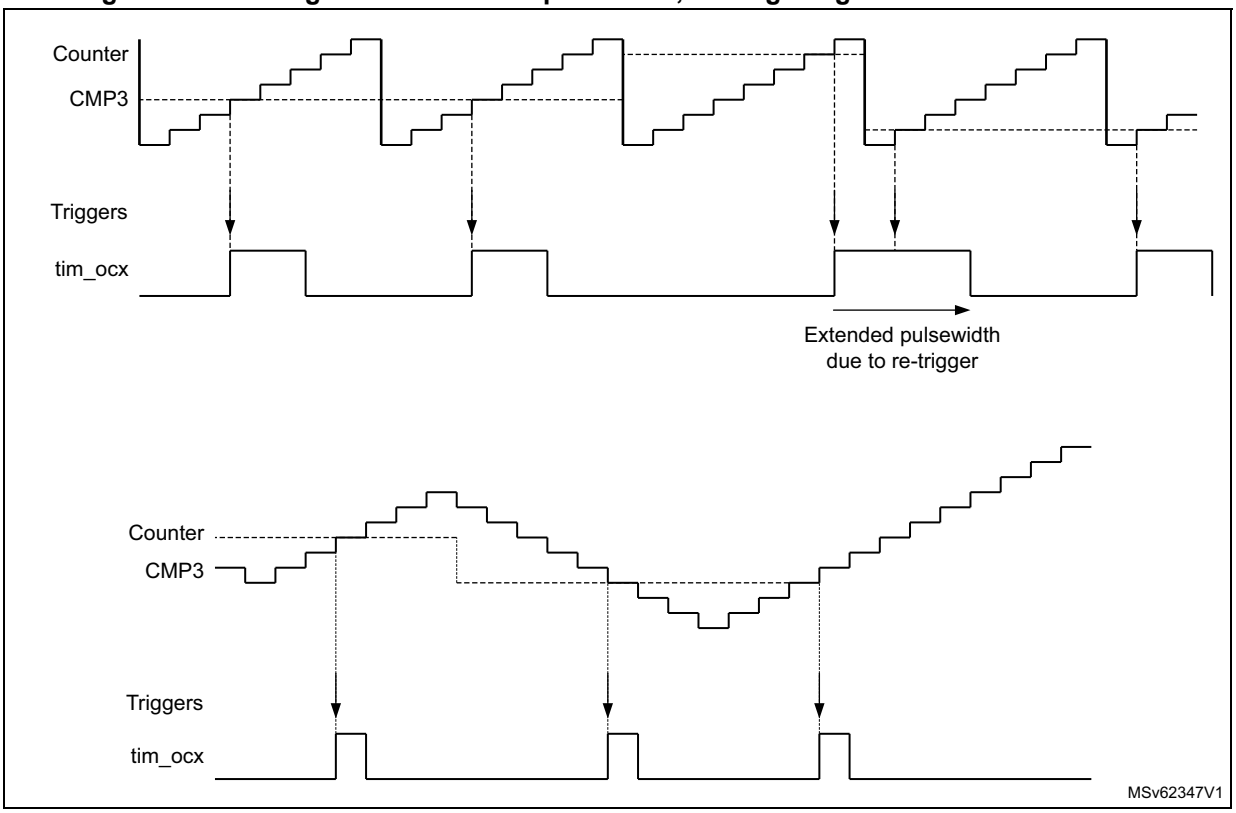


Figure 401 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 401. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

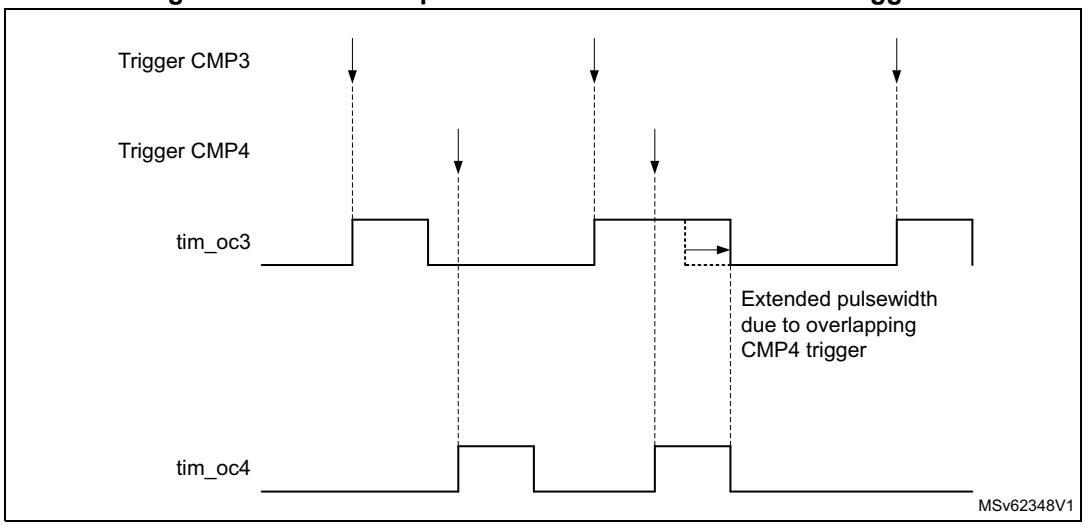
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 402).

Figure 402. Extended pulsewidth in case of concurrent triggers



38.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 407*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 407. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 403 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIMx_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, Counter enabled).

Figure 403. Example of counter operation in encoder interface mode.

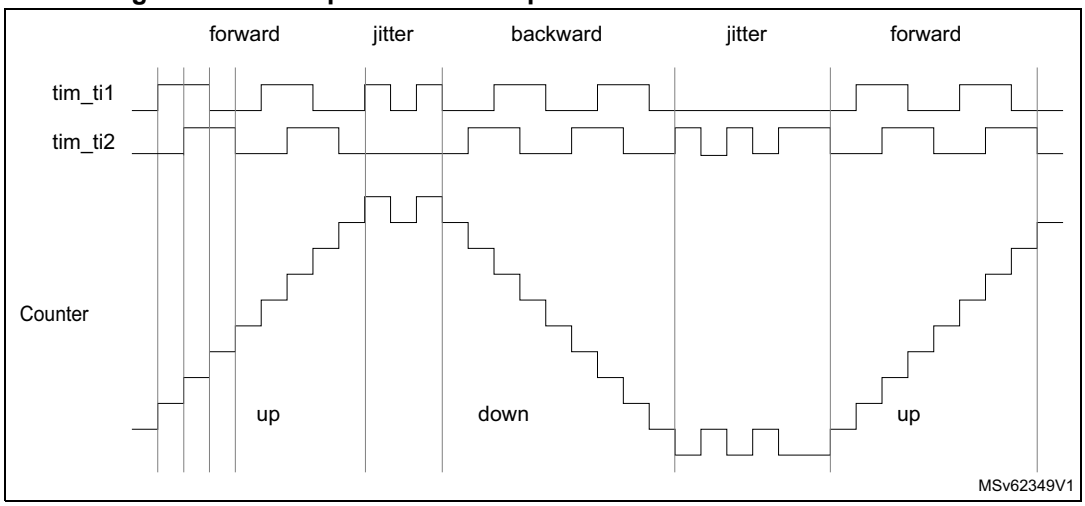


Figure 404 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 404. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

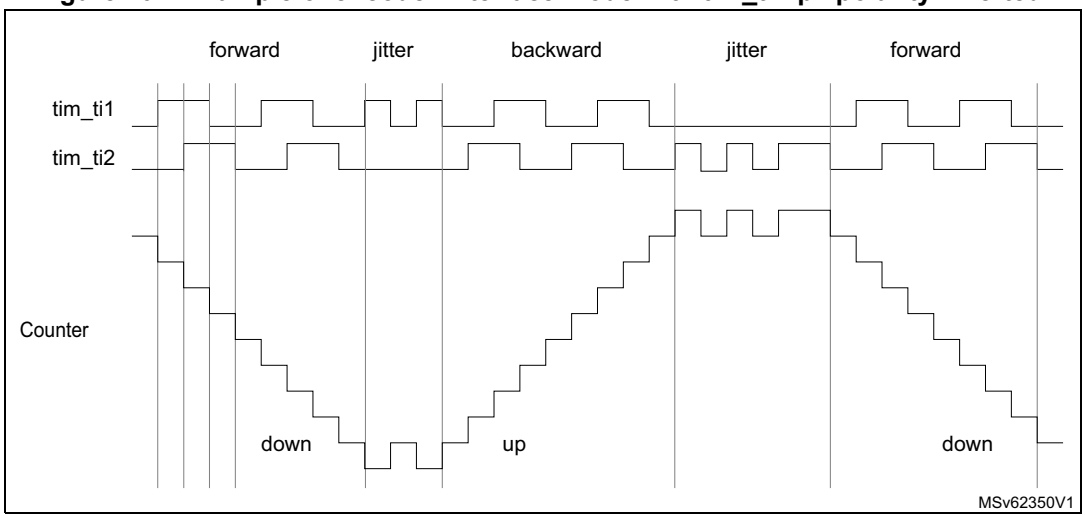
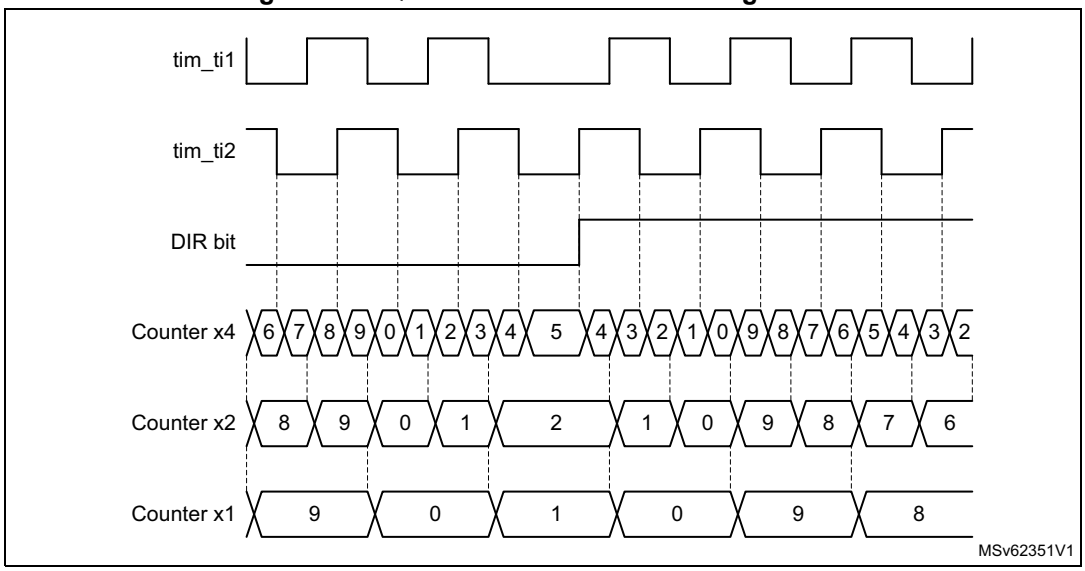


Figure 405 shows the timer counter value during a speed reversal, for various counting modes.

Figure 405. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

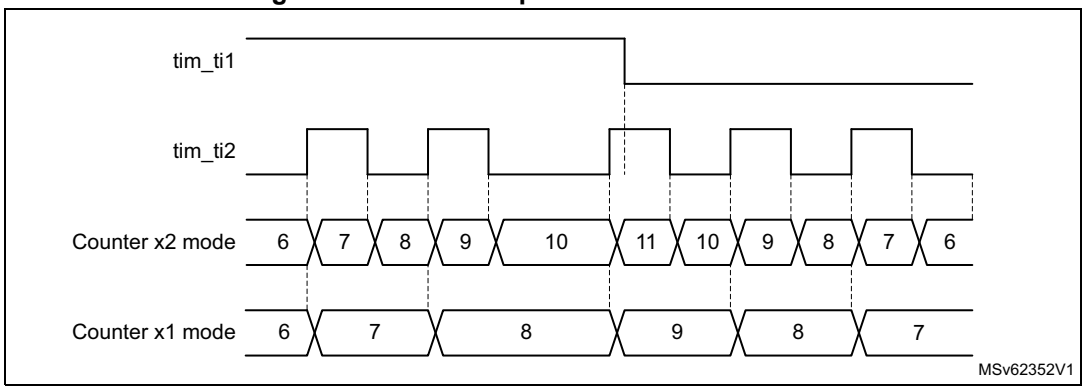
In the clock plus direction mode shown on *Figure 406*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 406. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 407*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 407*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 408*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 407*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 408*).

Figure 407. Directional clock encoder mode (CC1P = CC2P = 0)

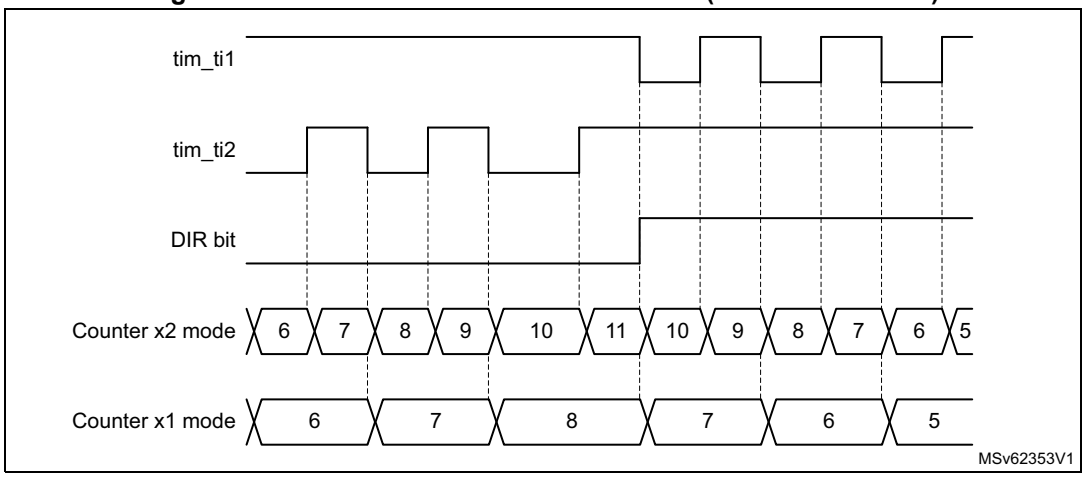


Figure 408. Directional clock encoder mode (CC1P = CC2P = 1)

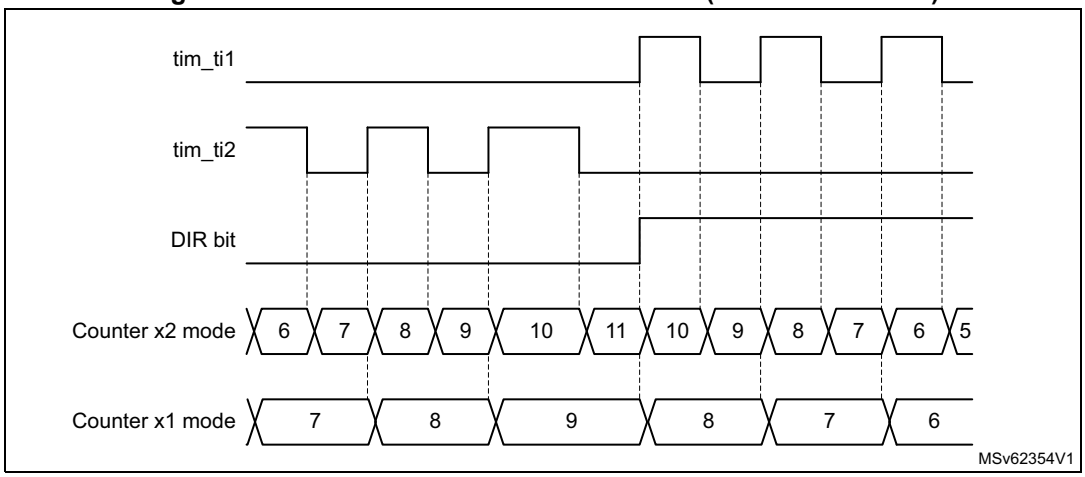


Table 408 here-below details how the directional clock mode operates, for any input transition.

Table 408. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

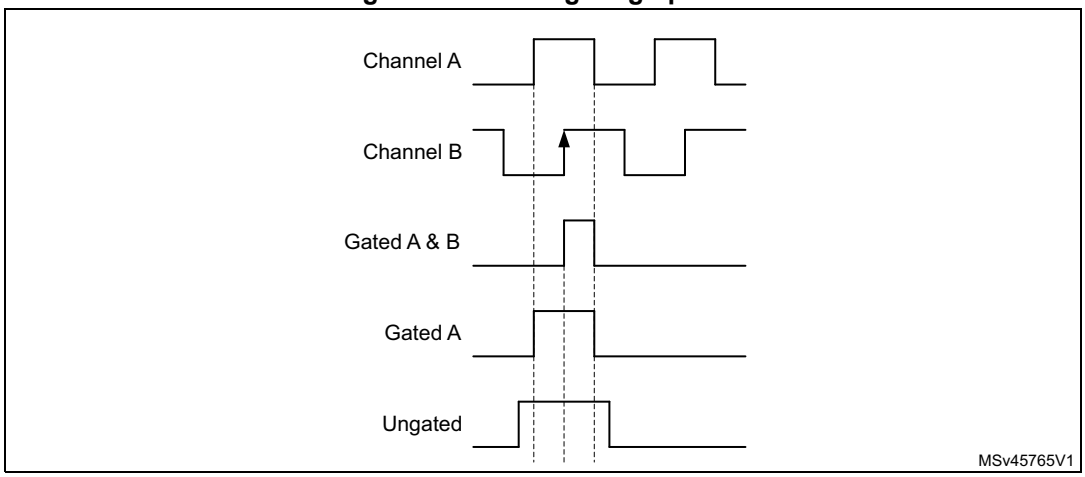
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 409*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

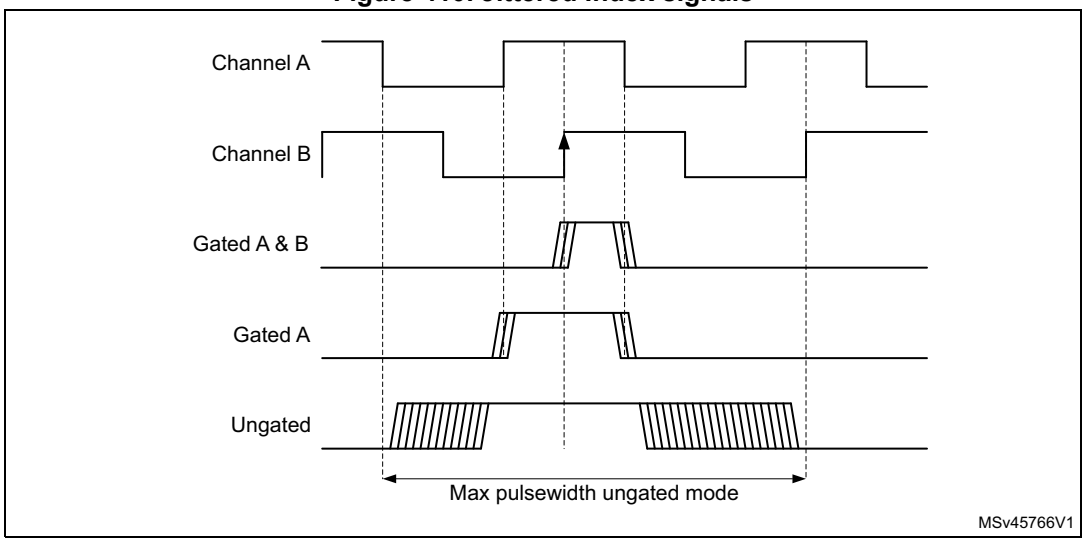
Figure 409. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 410*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 410. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 411* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 411. Index generation for IPOS[1:0] = 11

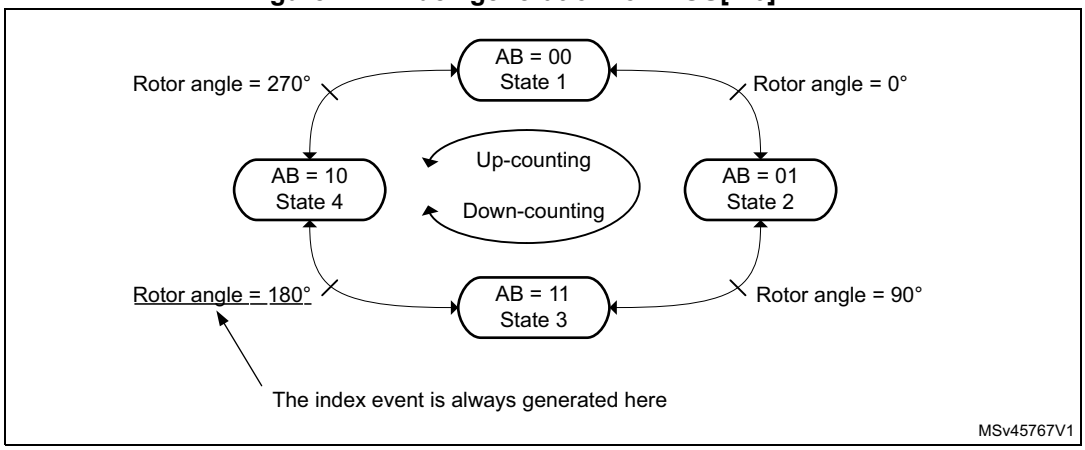


Figure 412 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 412. Counter reading with index gated on channel A (IPOS[1:0] = 11)

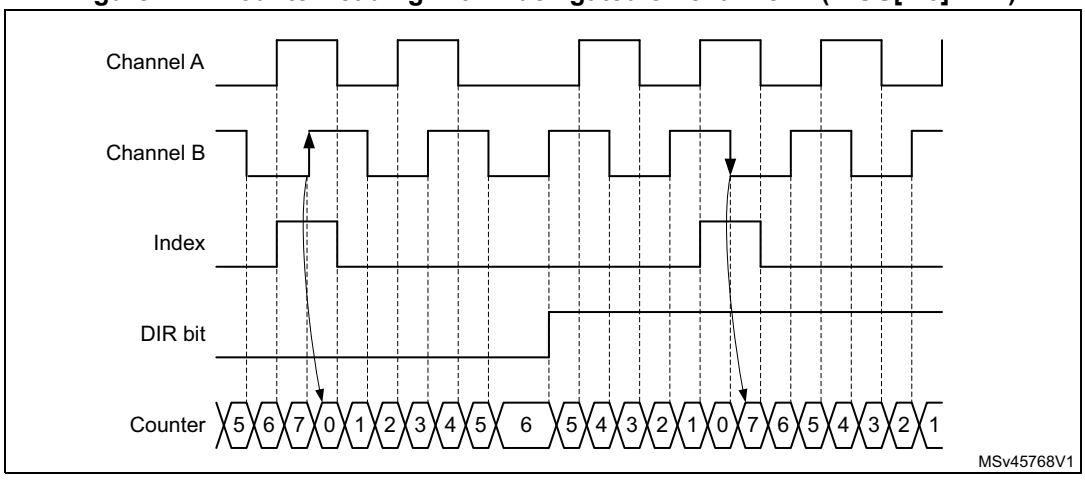


Figure 413. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 413. Counter reading with index ungated (IPOS[1:0] = 00)

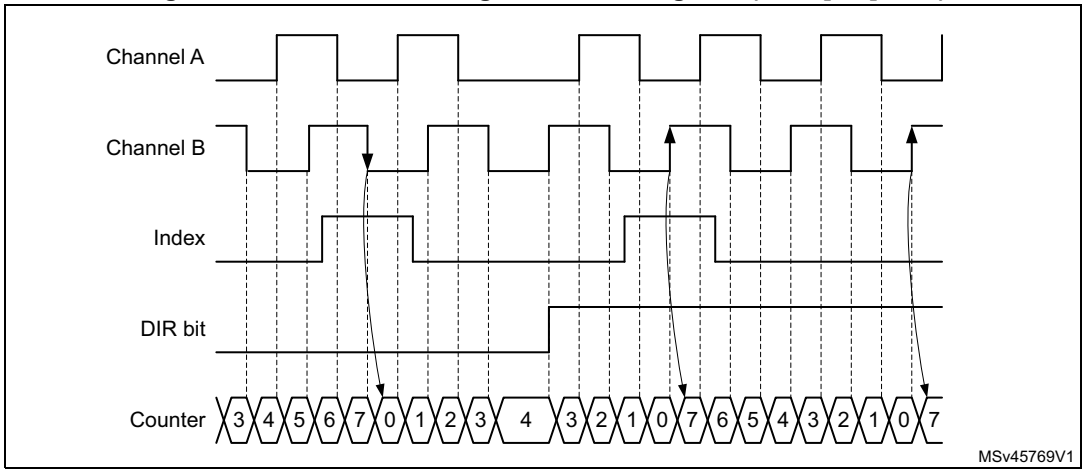


Figure 414. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 414. Counter reading with index gated on channel A and B

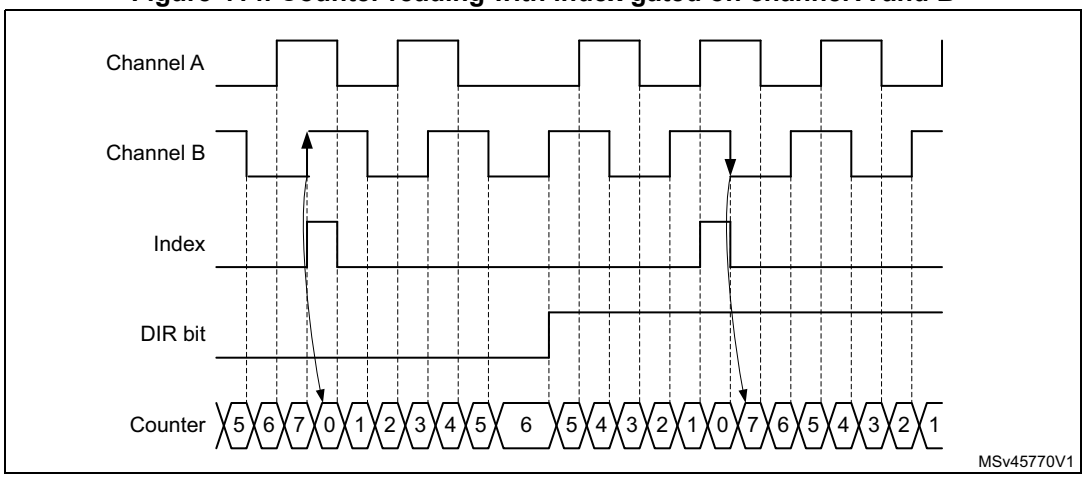


Figure 415 and *Figure 416* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 415. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

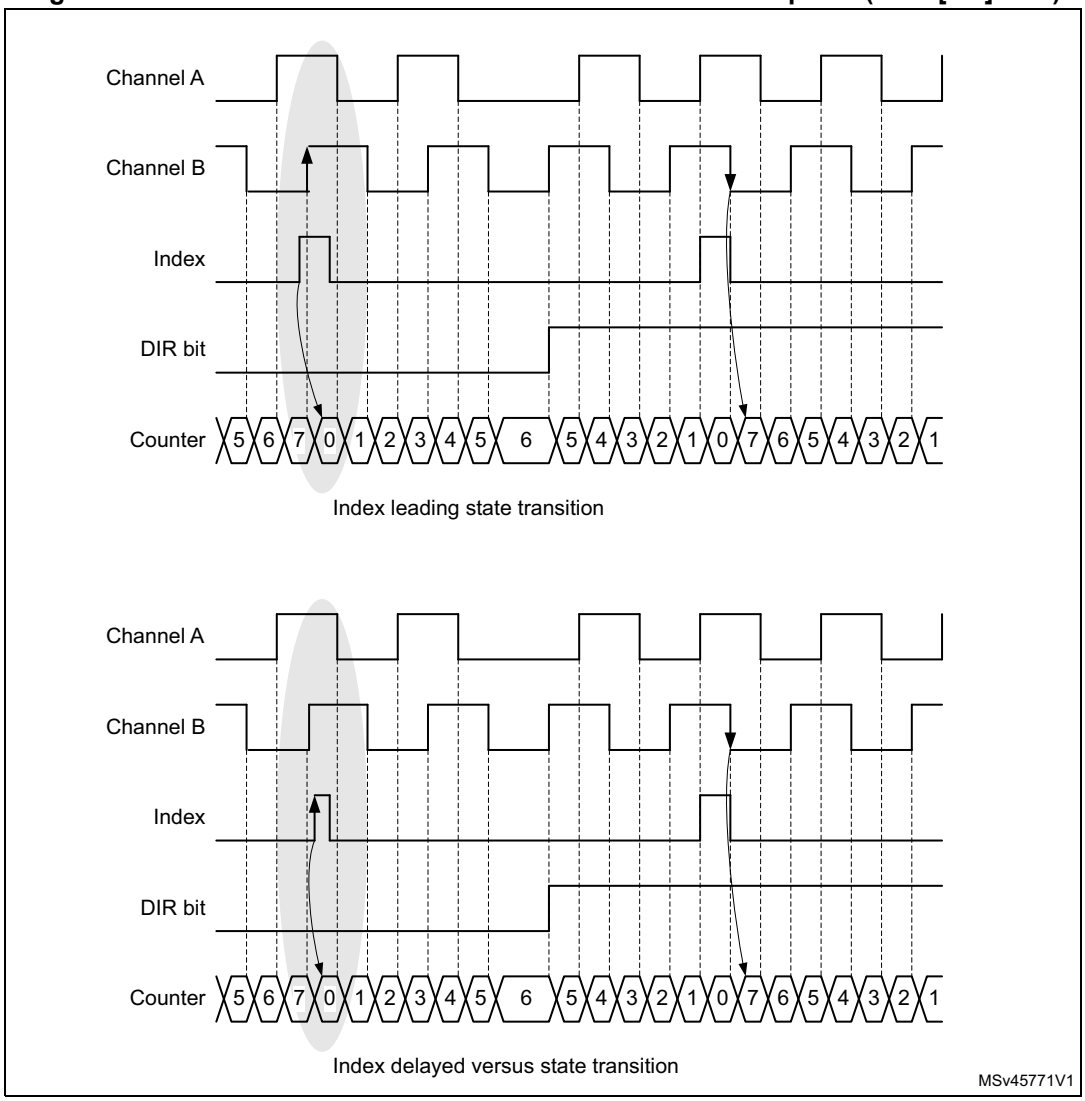


Figure 416. Counter reset Narrow index pulse (closer view, ARR = 0x07)

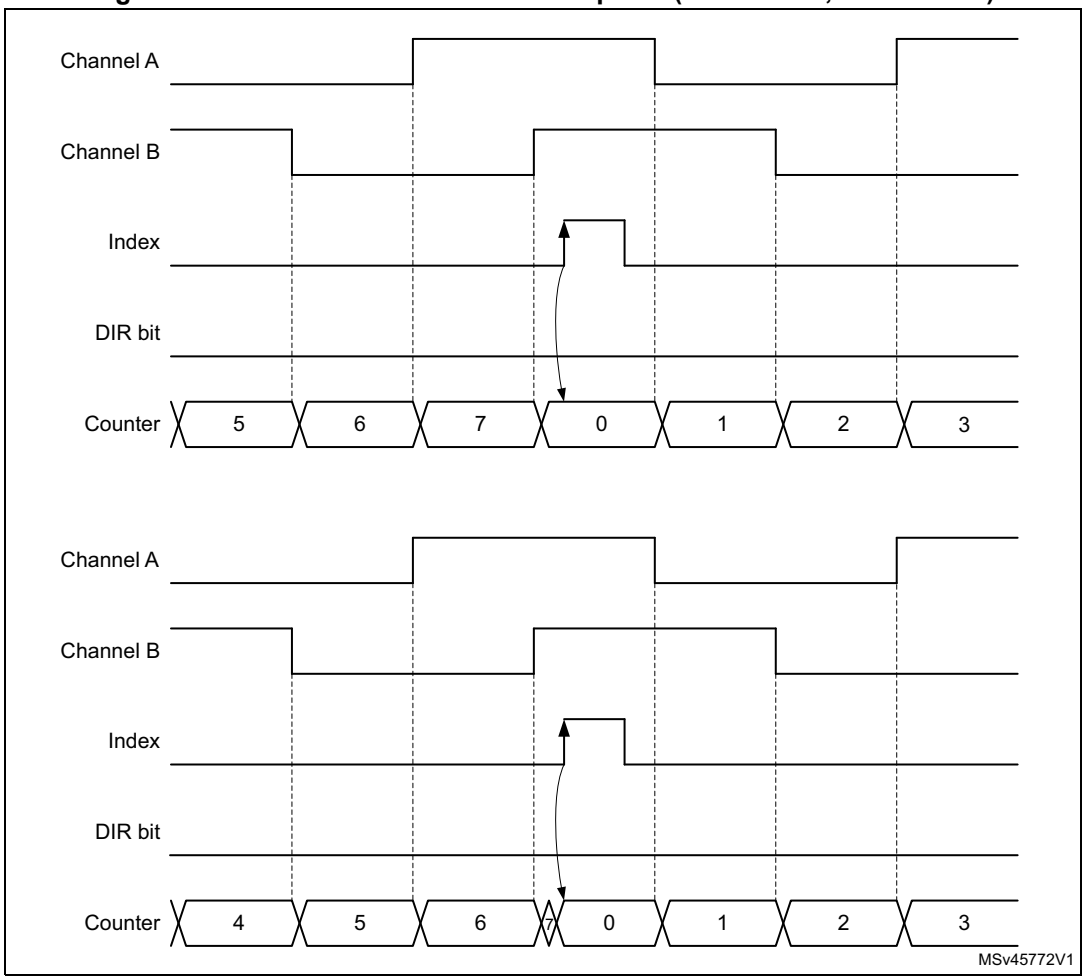
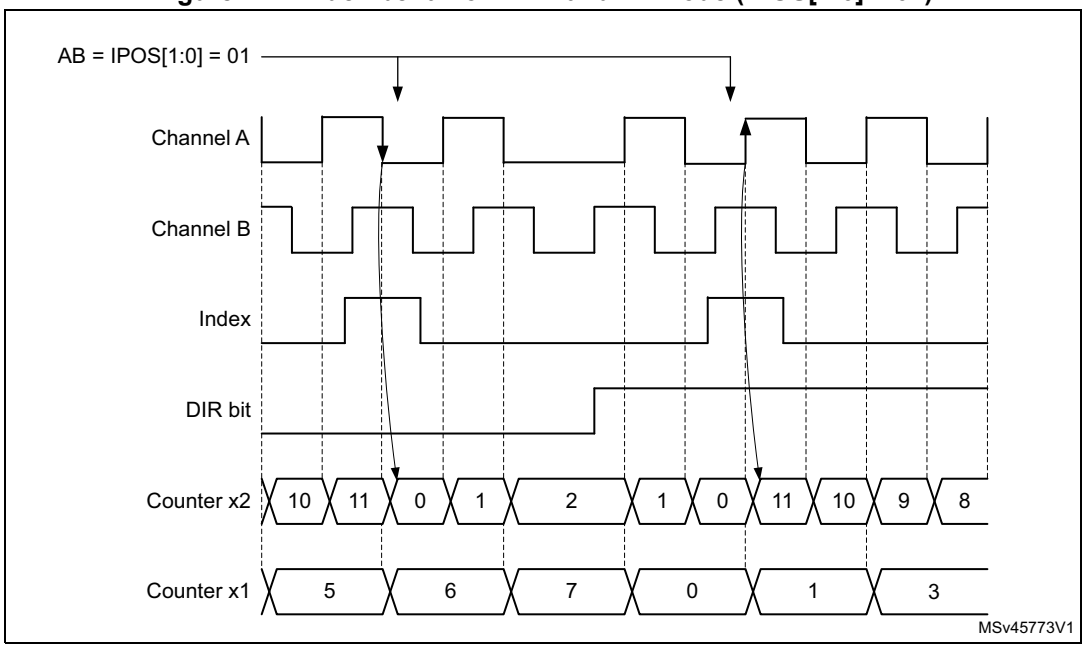


Figure 417 shows how the index is managed in x1 and x2 modes.

Figure 417. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

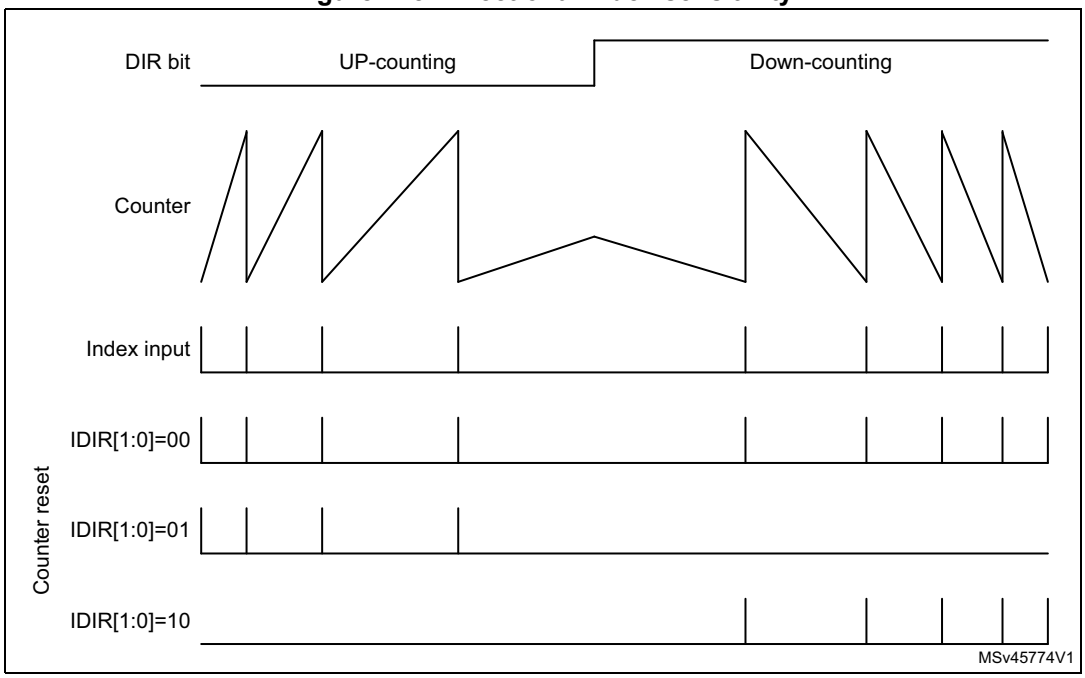


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 418 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

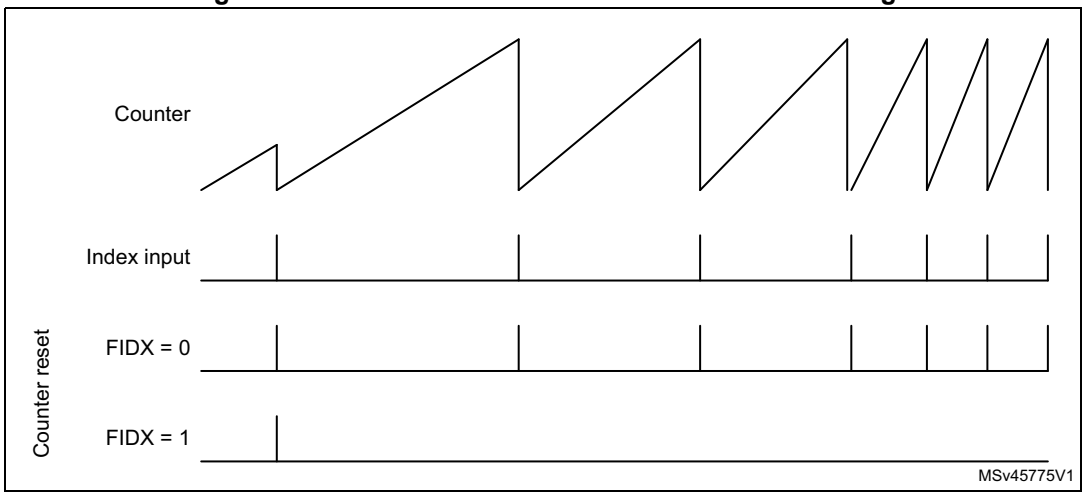
Figure 418. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 419*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 419. Counter reset as function of FIDX bit setting



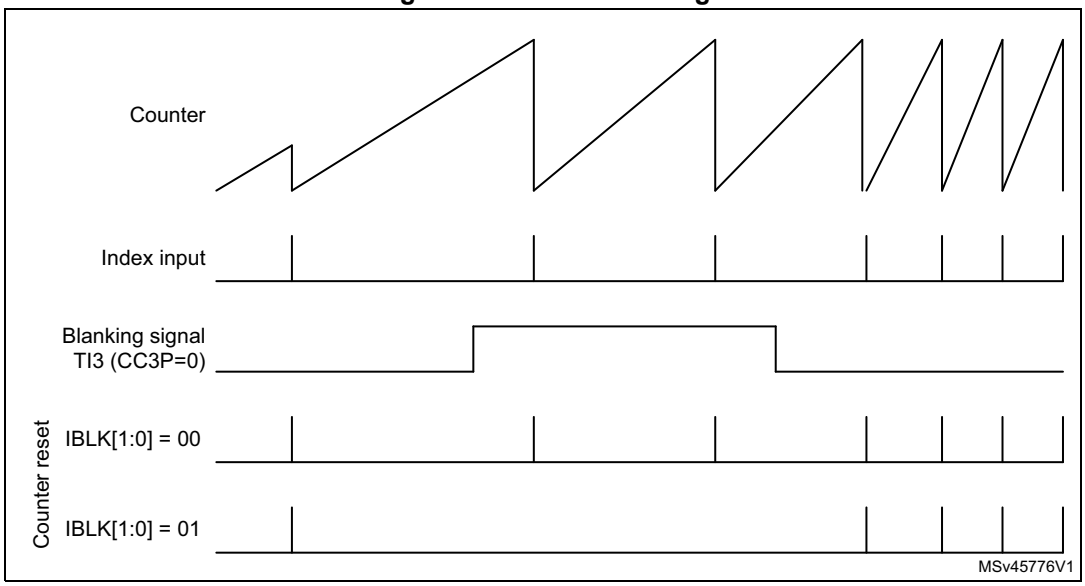
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 420* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 420. Index blanking



Index management in nonquadrature mode

Figure 421 and *Figure 422* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 421. Index behavior in clock + direction mode, IPOS[0] = 1

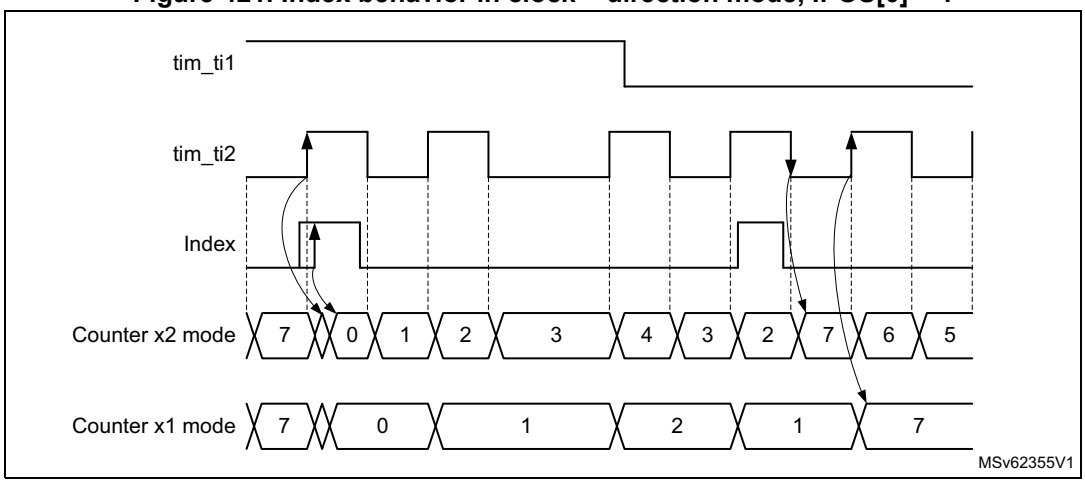
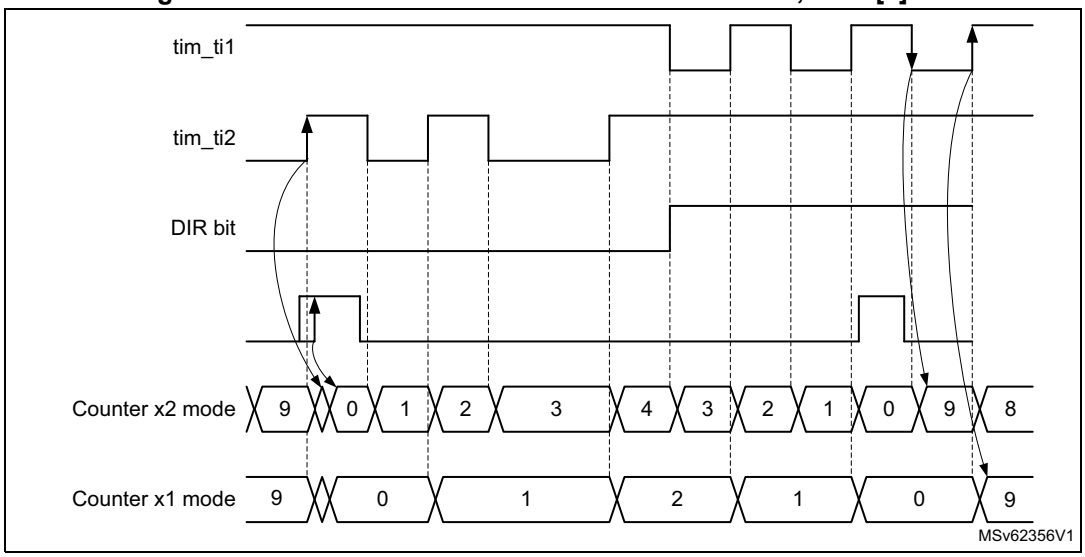


Figure 422. Index behavior in directional clock mode, IPOS[0] = 1

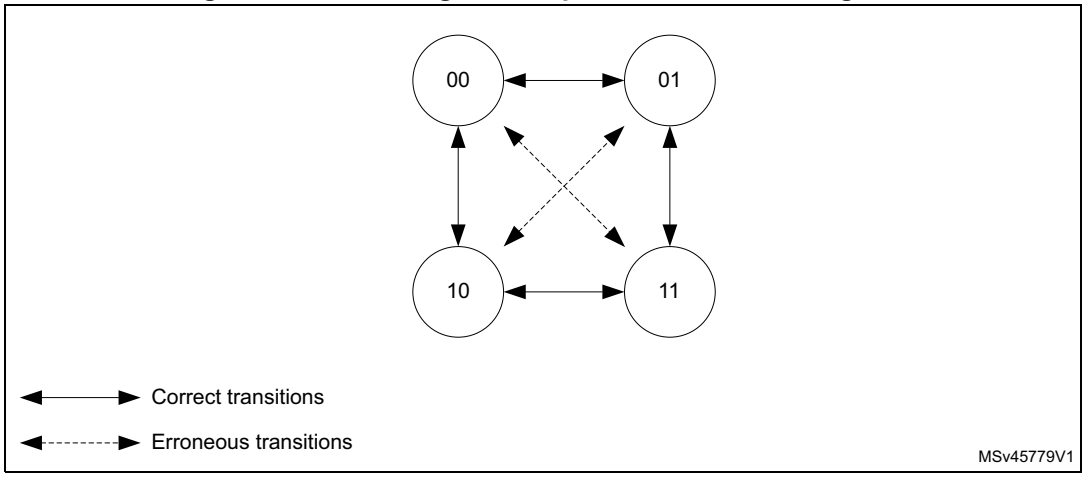


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 423.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 423. State diagram for quadrature encoded signals



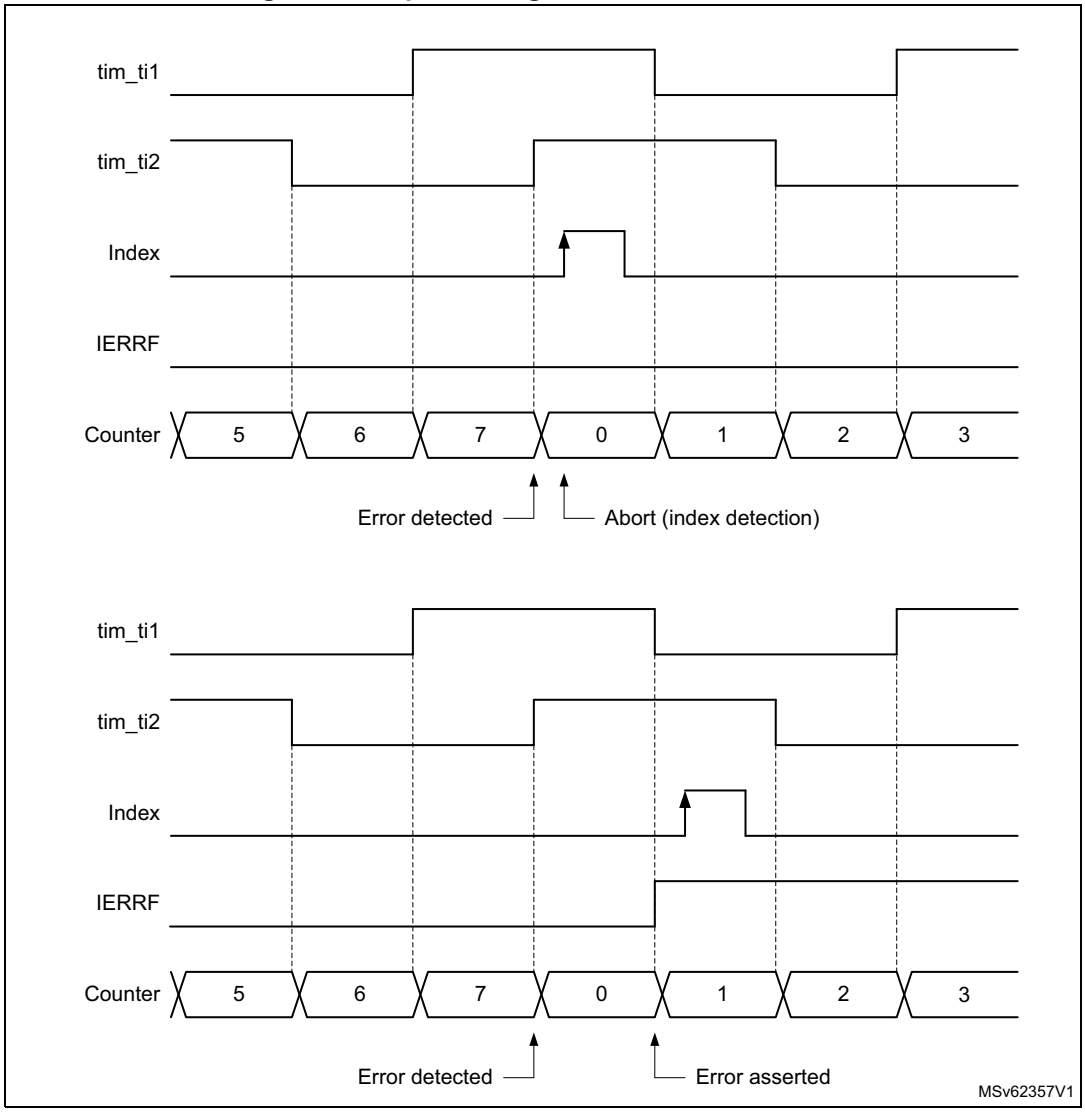
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

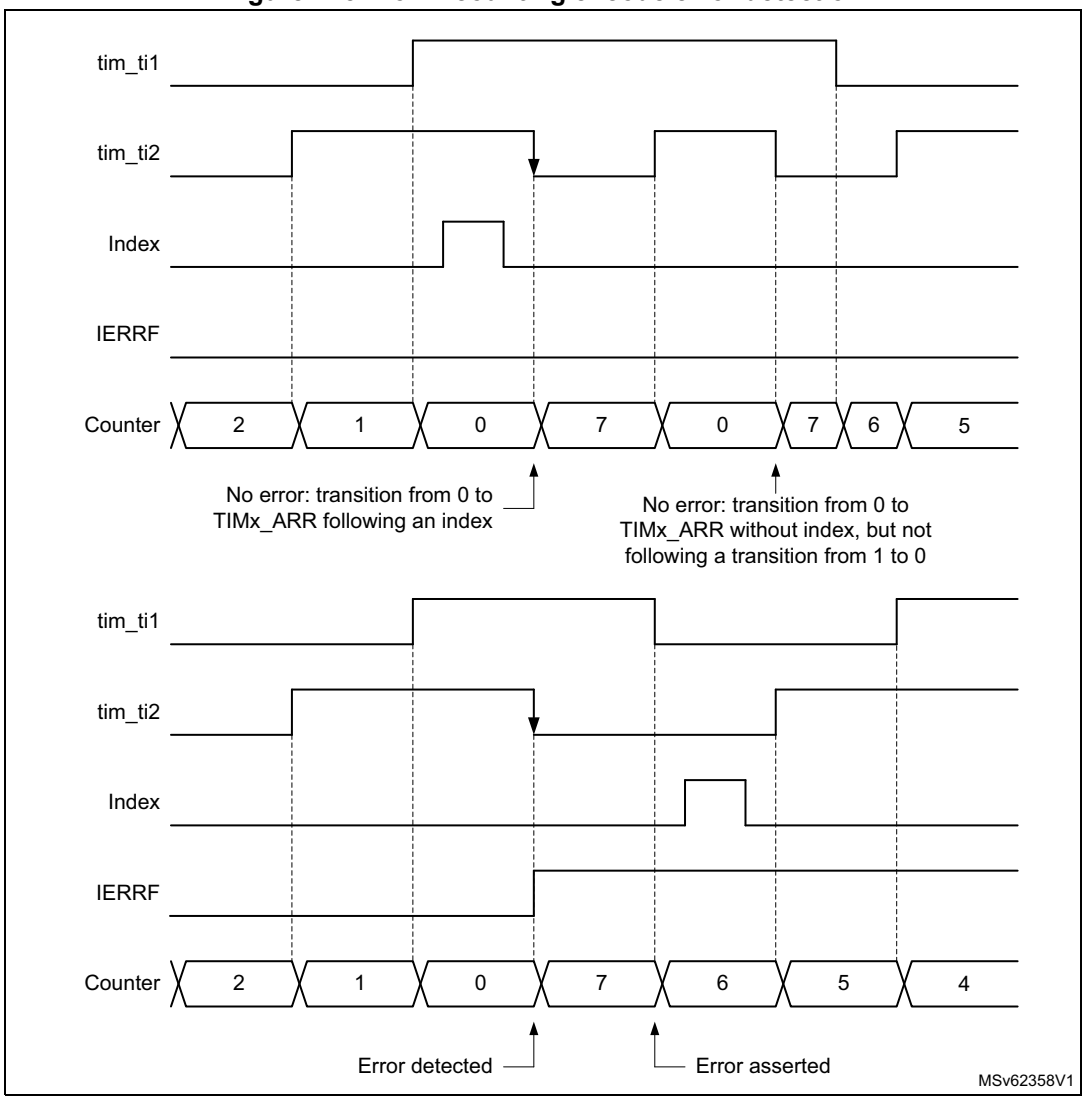
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 424*.

Figure 424. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 425*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 425. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

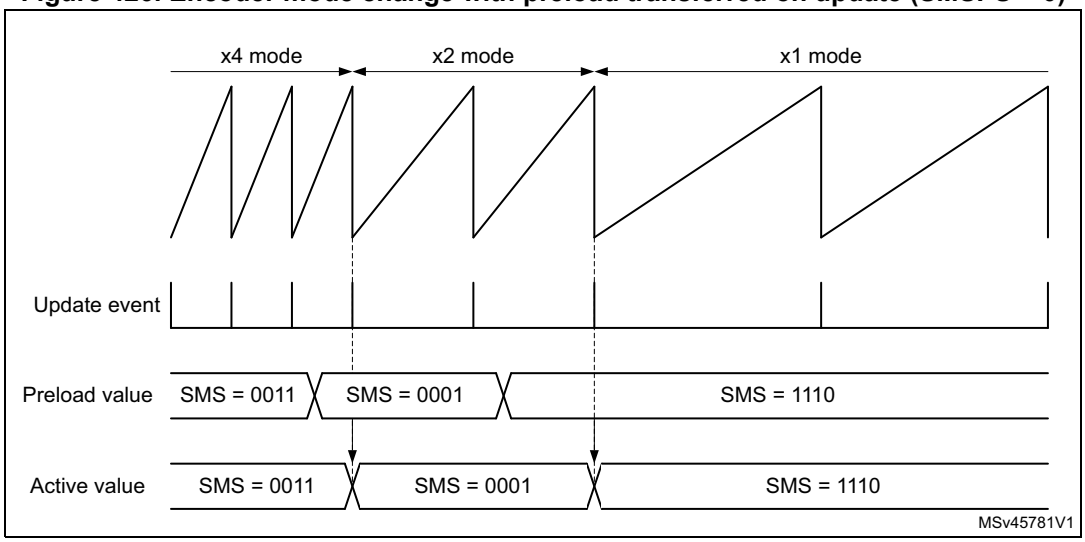
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 426*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 426. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

38.3.26 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

38.3.27 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

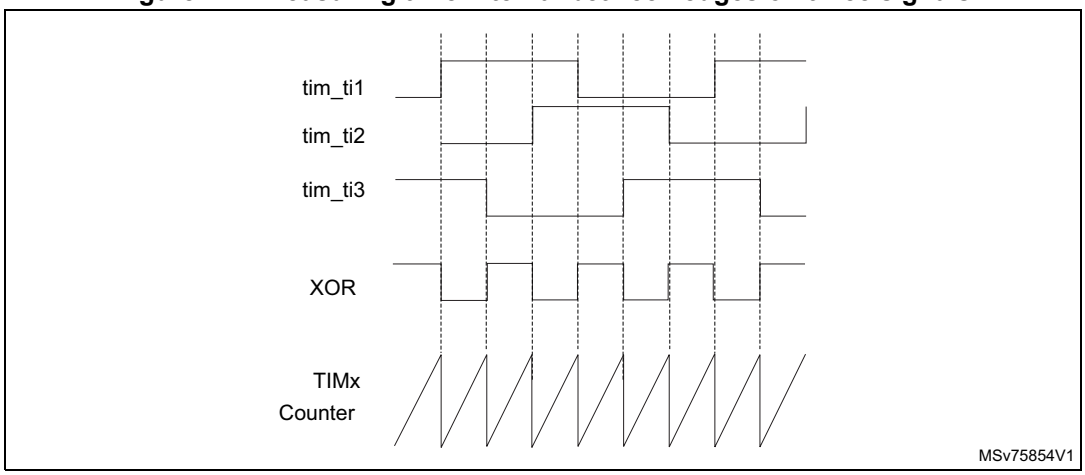
There is no latency between the UIF and UIFCPY flags assertion.

38.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 427*.

Figure 427. Measuring time interval between edges on three signals



38.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 428*. The "interfacing timer" captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 370*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

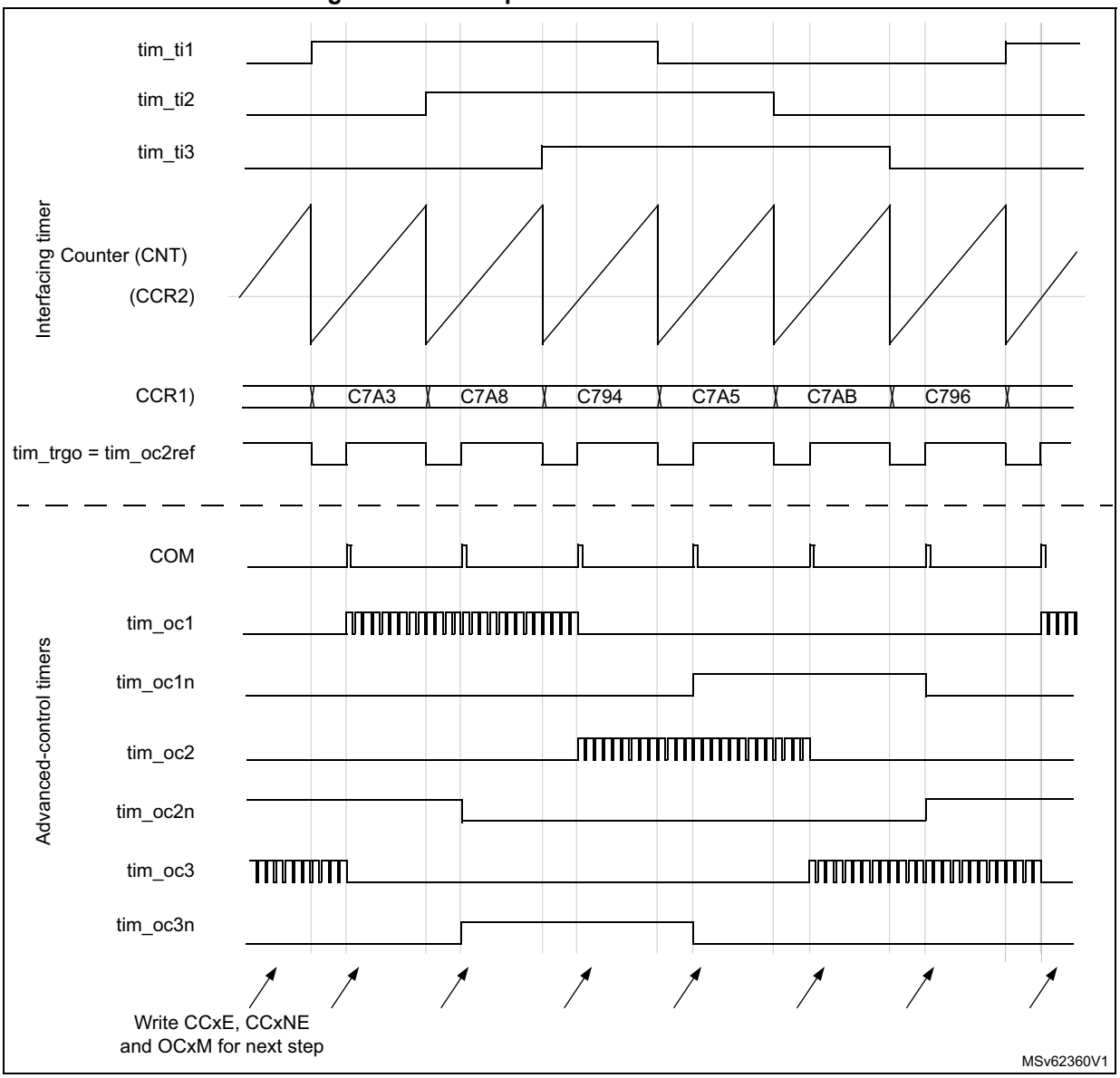
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to 1.
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIMx_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 428 describes this example.

Figure 428. Example of Hall sensor interface



38.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 39.4.23: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

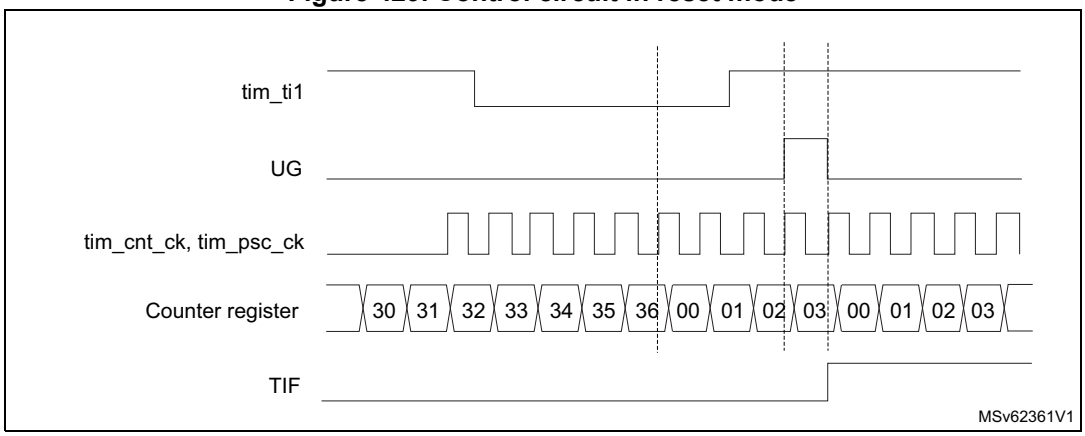
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 429. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

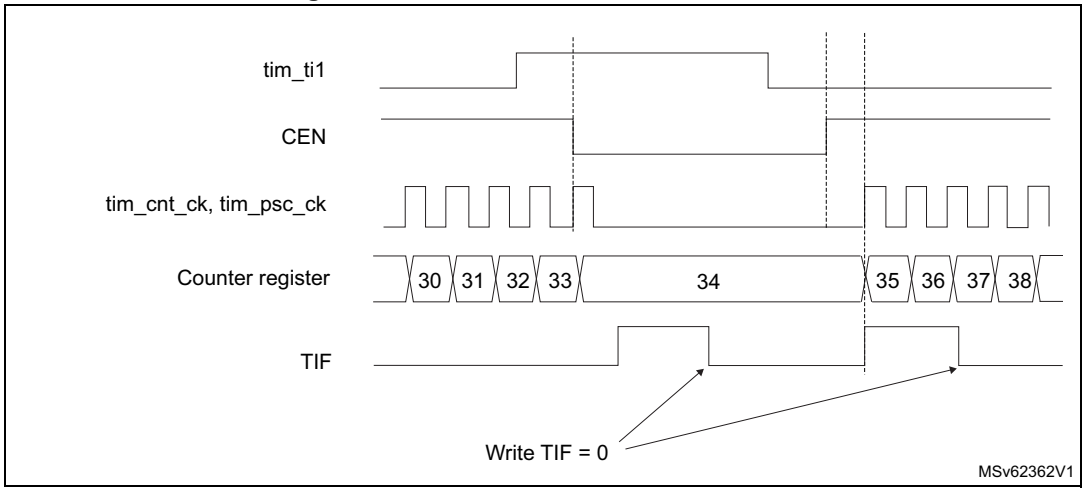
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 430. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

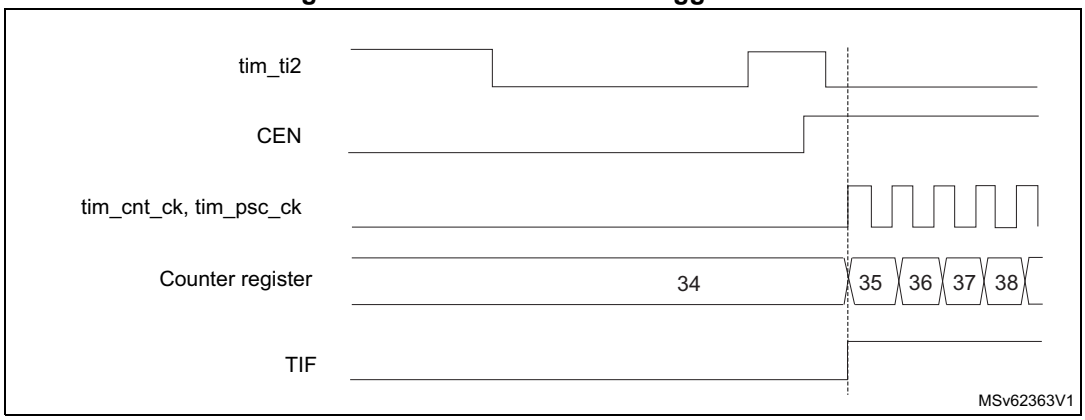
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 431. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

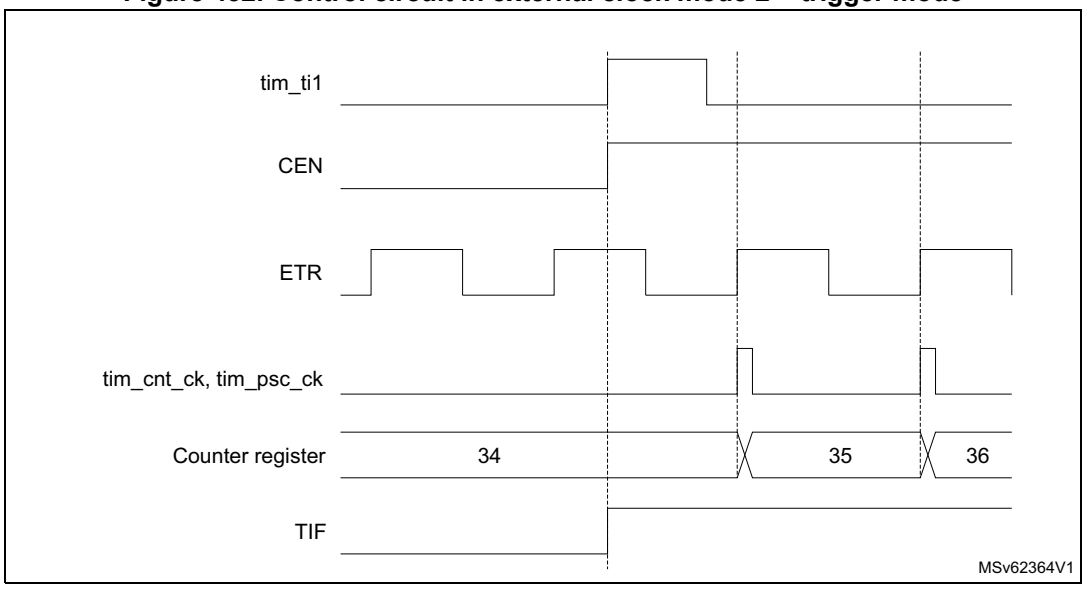
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 432. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 385*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

38.3.32 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 38.6.29: TIMx DMA control register (TIMx_DCR)(x = 1, 8)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address.
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (see note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

38.3.33 TIM1/TIM8 DMA requests

The TIM1/TIM8 can generate a DMA request, as shown in the table below.

Table 409. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

38.3.34 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

38.4 TIM1/TIM8 low-power modes

Table 410. Effect of low-power modes on TIM1/TIM8

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

38.5 TIM1/TIM8 interrupts

The TIM1/TIM8 can generate multiple interrupts, as shown in *Table 411*.

Table 411. Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD		Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC		Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
		Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
		Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
		Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI_COM_DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_IERR	TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
		Break2	B2IF		write 0 in B2IF	Yes	No
		System Break	SBIF		write 0 in SBIF	Yes	No
	TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
	TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

38.6 TIM1/TIM8 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

38.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

38.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
						rw		rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

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

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 (after a dead-time) when MOE = 0
1: tim_oc1 = 1 (after a dead-time) when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

38.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Table 398: Internal trigger connection* for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS:** OCREF clear selection

This bit is used to select the OCREF clear source.

0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Bits 16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

38.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

38.6.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF
								rc_w0	rc_w0	rc_w0	rc_w0			rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

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

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 38.6.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

38.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation

This bit can be set by software, it is automatically cleared by hardware

0: No action

1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)

Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation

Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation

Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (downcounting).

38.6.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

38.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

38.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

38.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

38.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 412* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 412. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CCxE bit</th><th>CCxNE bit</th><th>tim_ocx output state</th><th>tim_ocxn output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0, tim_ocxn = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output disabled (not driven by the timer: Hi-Z)
tim_ocx = 0</td><td>tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_ocxn = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>OCREF + Polarity + dead-time</td><td>Complementary to OCREF (not OCREF) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_ocx = CCxP</td><td>tim_ocxref + Polarity
tim_ocxn = tim_ocxref x or CCxNP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_ocxref + Polarity
tim_ocx = tim_ocxref xor CCxP</td><td>Off-State (output enabled with inactive state)
tim_ocxn = CCxNP</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z).</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered).
Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration).
Note: tim_brk2 can only be used if OSSI = OSSR = 1.</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

38.6.12 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

38.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

38.6.14 TIMx autoreload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 38.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

38.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x030

Reset value: 0x0000

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

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

38.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

38.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

38.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

38.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

38.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection, see *Figure 390: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (*Section 38.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 390: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 38.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 38.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.
DTG[7:5] = 110 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.
DTG[7:5] = 111 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.
Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

38.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3
Distortion on channel 3 output:
0: No effect of tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2
Distortion on channel 2 output:
0: No effect of tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1
Distortion on channel 1 output:
0: No effect of oc5ref on oc1refc
1: oc1refc is the logical AND of oc1ref and oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: Capture/compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

38.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Capture/compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc6 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

38.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

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

38.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

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

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx => DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

38.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8)

Address offset: 0x058

Reset value: 0x0000 0000

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking
This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input
00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction
This bit indicates in which direction the Index event resets the counter.
00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable
This bit indicates if the Index event resets the counter.
0: Index disabled
1: Index enabled

38.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

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

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

38.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIMx_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

38.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

...

111: tim_ocref_clr7

Refer to *Section 38.3.2: TIM1/TIM8 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

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

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 38.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

38.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x3DC

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...

38.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

38.6.31 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 413. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value							0		0	0	0	0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	**TIMx_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP S [1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value									0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0

Table 413. TIMx register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIMx_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 413. TIMx register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	TIMx_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	TIMx_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIMx_CCMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL [2:0]			Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value														0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

39 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

39.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 39.4.23: Timer synchronization*.

39.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.

39.3 TIM2/TIM3/TIM4/TIM5 implementation

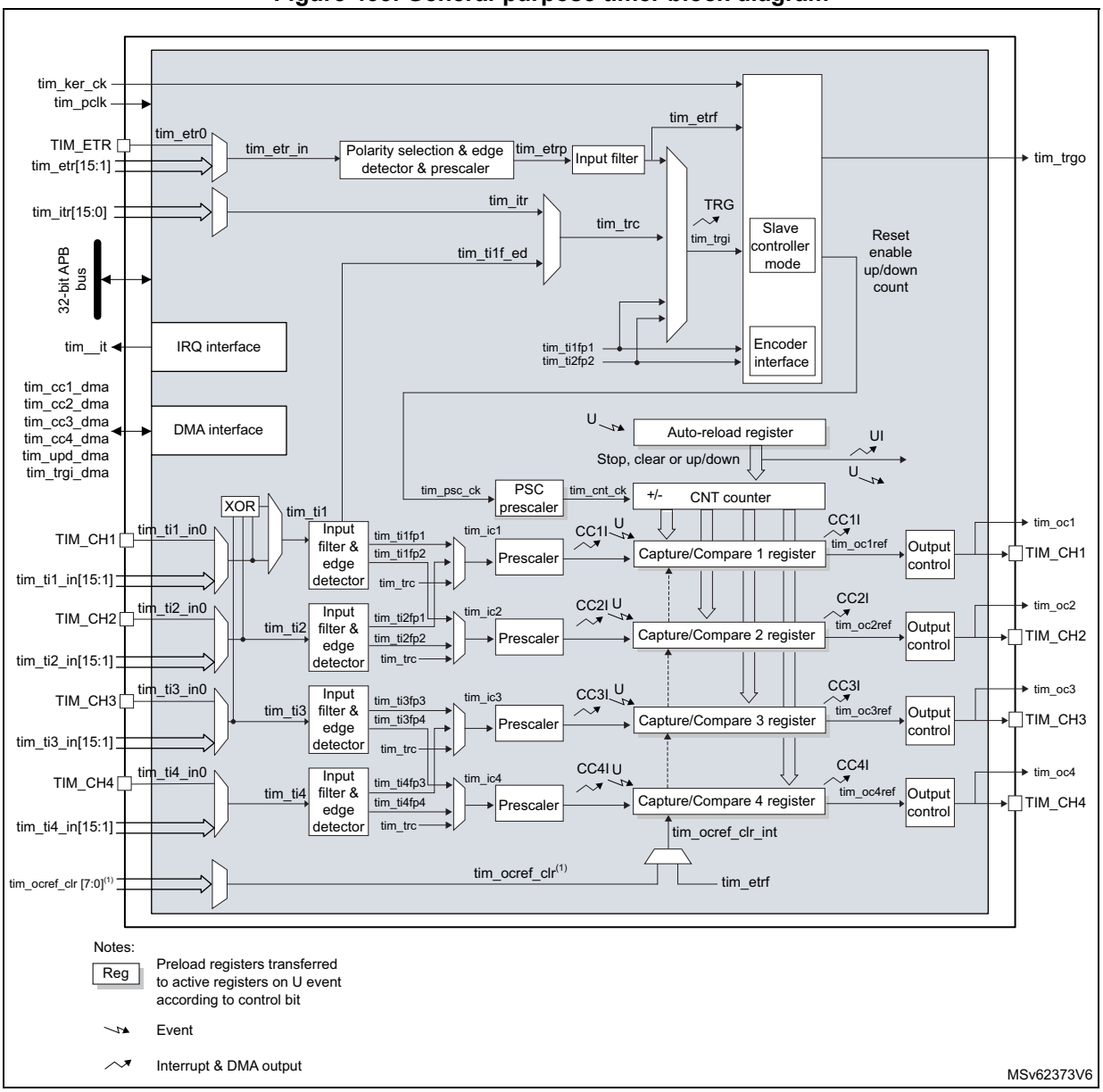
Table 414. STM32H5 general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	16-bit	16-bit	32-bit
OCREF clear selection	Yes	Yes	Yes	Yes
Sources	tim_etrf Reserved	tim_etrf Reserved	tim_etrf Reserved	tim_etrf Reserved

39.4 TIM2/TIM3/TIM4/TIM5 functional description

39.4.1 Block diagram

Figure 433. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 39.3: TIM2/TIM3/TIM4/TIM5 implementation*.

39.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 415 and *Table 416* in this section summarize the TIM inputs and outputs.

Table 415. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 416. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 416. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.

Table 417, *Table 418*, *Table 419* and *Table 420* are listing the sources connected to the tim_ti[4:1] input multiplexers.

Table 417. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	eth_ptp_pps_o	eth_ptp_pps_o	Reserved	Reserved
tim_ti1_in[15:2]	Reserved			

Table 418. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in[15:1]	Reserved			

Table 419. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in[15:1]	Reserved			

Table 420. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in[15:1]	Reserved			

Table 421 lists the internal sources connected to the tim_etr input multiplexer.

Table 421. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	Reserved	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	Reserved	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo	tim5_trgo	Reserved
tim_itr5	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr6	tim12_trgo	tim12_trgo	tim12_trgo	tim12_trgo
tim_itr7	tim13_oc1	tim13_oc1	tim13_oc1	tim13_oc1
tim_itr8	tim14_oc1	tim14_oc1	tim14_oc1	tim14_oc1
tim_itr9	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr10	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr11	tim17_oc1	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr12	USBSOF	Reserved	Reserved	USBSOF
tim_itr[15:13]	Reserved			

Table 422 lists the internal sources connected to the tim_etr input multiplexer.

Table 422. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	Reserved	Reserved	Reserved	sai2_fs_a
tim_etr2				sai2_fs_b

Table 422. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signals assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr3	RCC_LSE	Reserved	Reserved	Reserved
tim_etr4	sai1_fs_a			
tim_etr5	sai1_fs_b			
tim_etr6	Reserved			
tim_etr7				
tim_etr8		TIM2_ETR	TIM2_ETR	TIM2_ETR
tim_etr9	TIM3_ETR	Reserved	TIM3_ETR	TIM3_ETR
tim_etr10	TIM4_ETR	TIM4_ETR	Reserved	TIM4_ETR
tim_etr11	TIM5_ETR	TIM5_ETR	TIM5_ETR	Reserved
tim_etr12	Reserved	Reserved	Reserved	Reserved
tim_etr13				
tim_etr14	eth_ptp_pps_o	eth_ptp_pps_o		
tim_etr15	Reserved			

39.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set one clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 434 and *Figure 435* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 434. Counter timing diagram with prescaler division change from 1 to 2

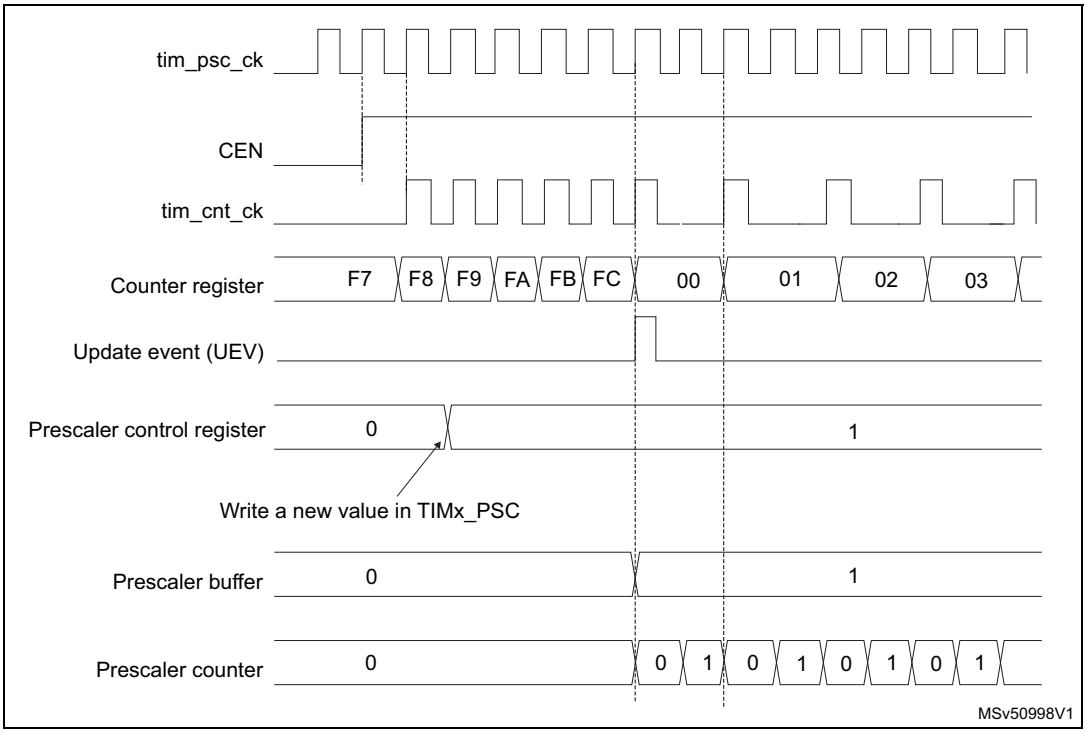
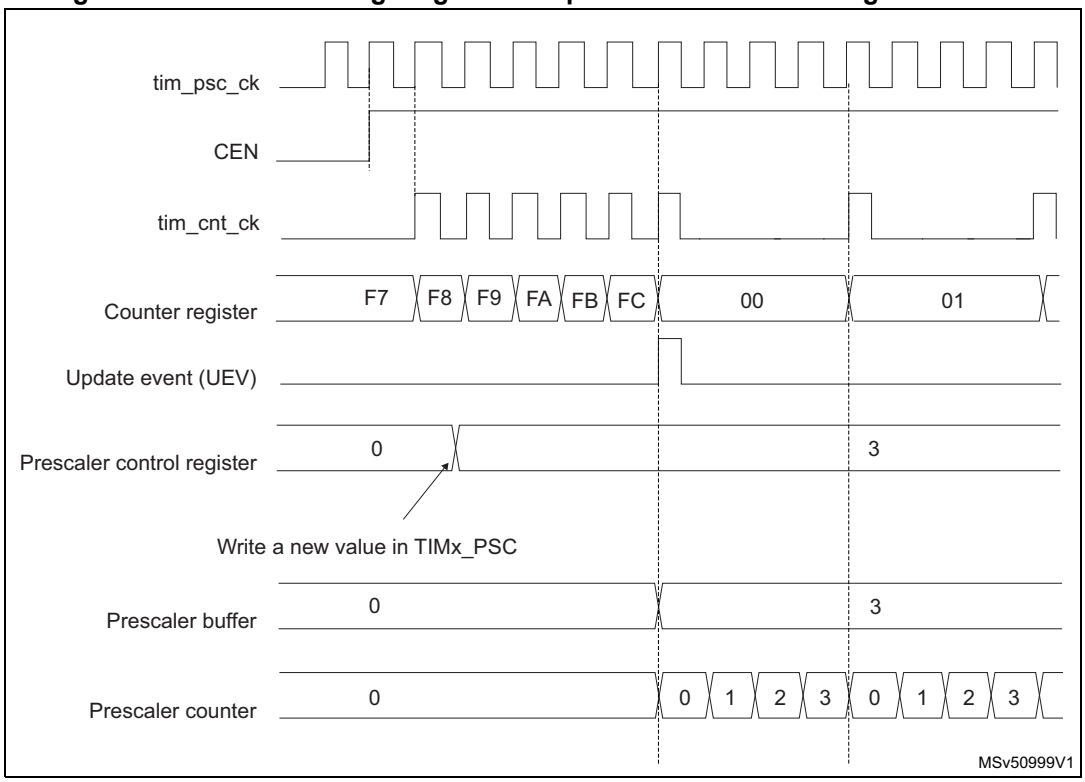


Figure 435. Counter timing diagram with prescaler division change from 1 to 4



39.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 436. Counter timing diagram, internal clock divided by 1

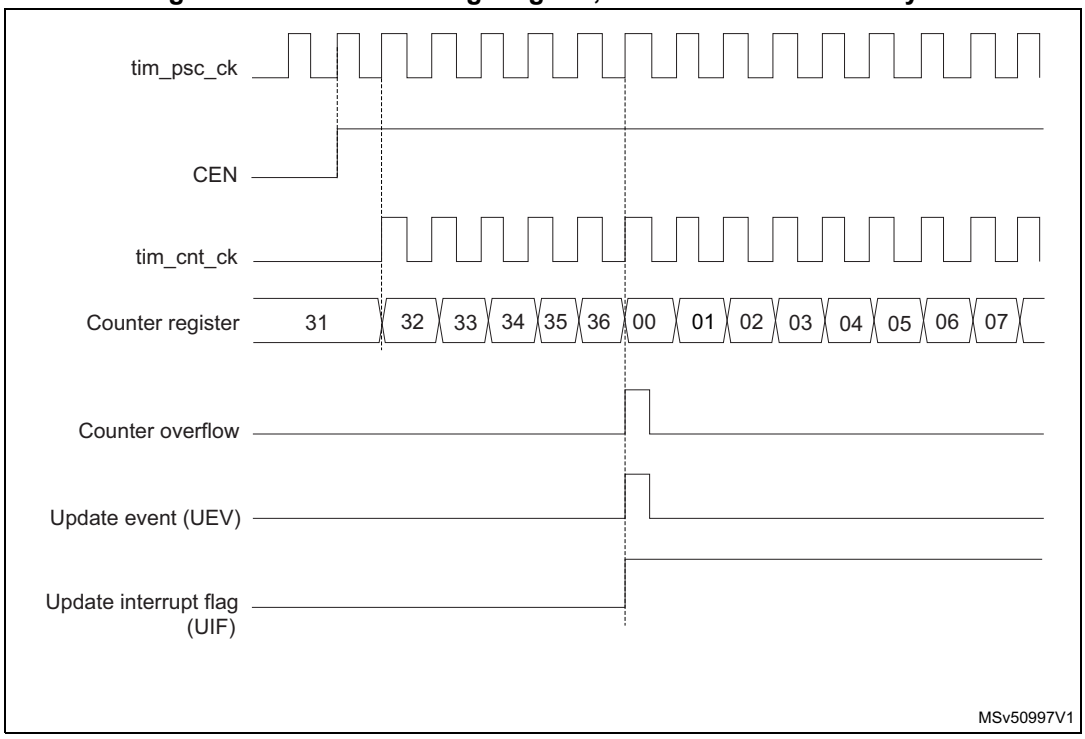


Figure 437. Counter timing diagram, internal clock divided by 2

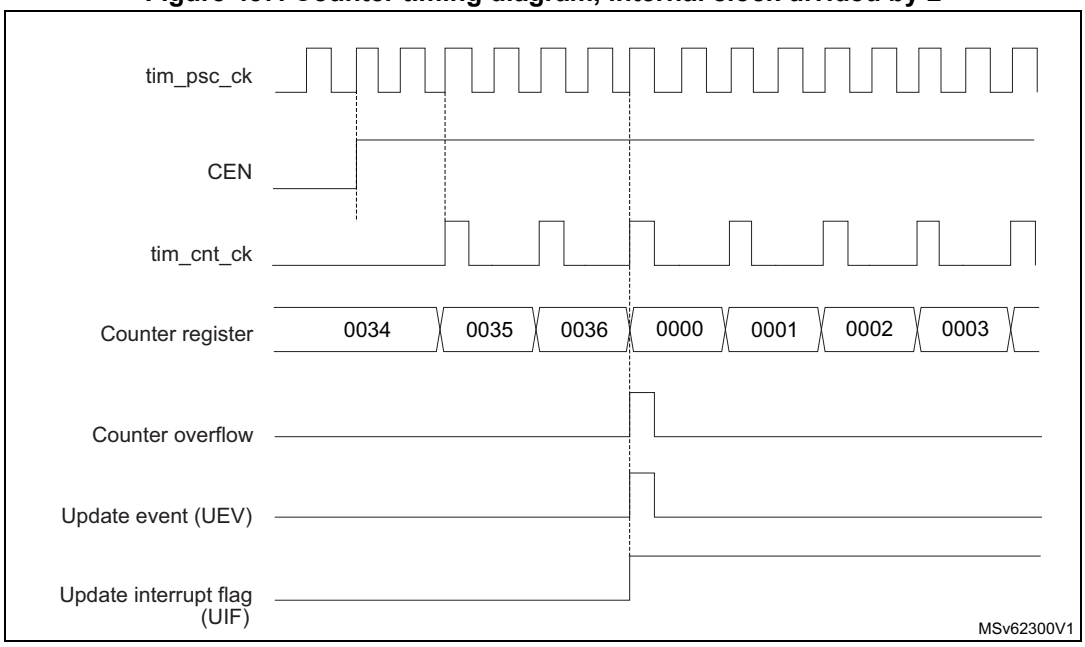


Figure 438. Counter timing diagram, internal clock divided by 4

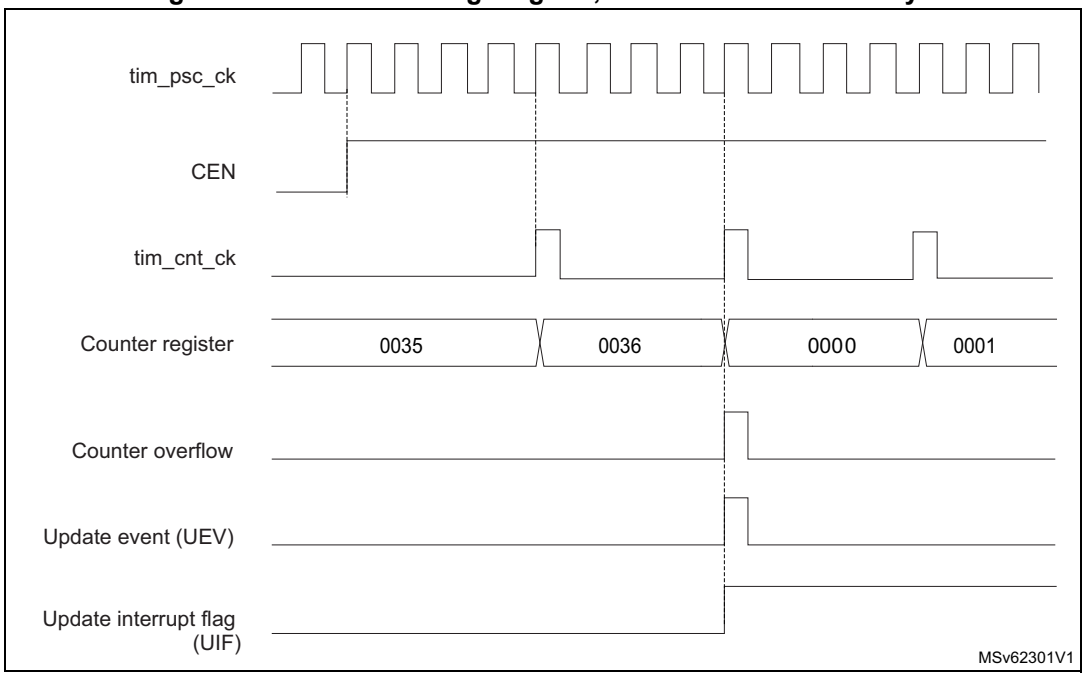


Figure 439. Counter timing diagram, internal clock divided by N

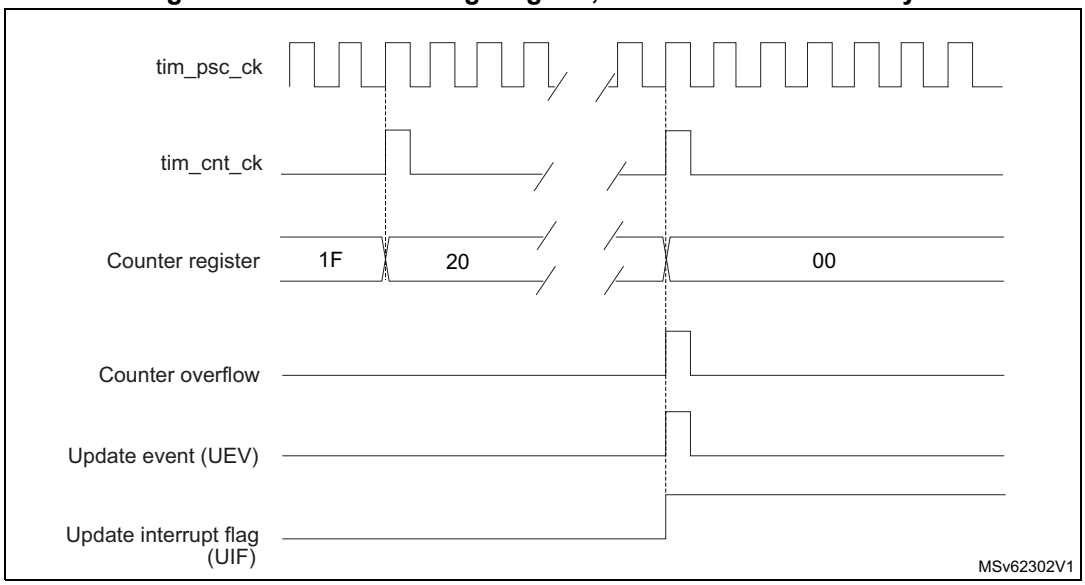


Figure 440. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

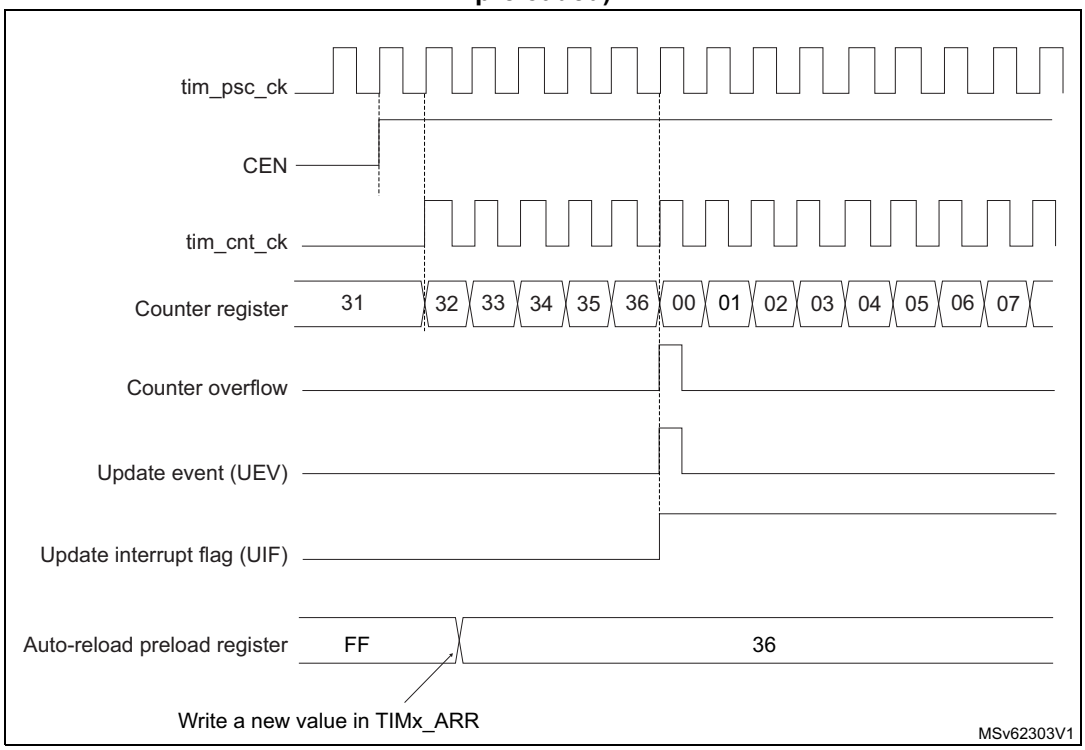
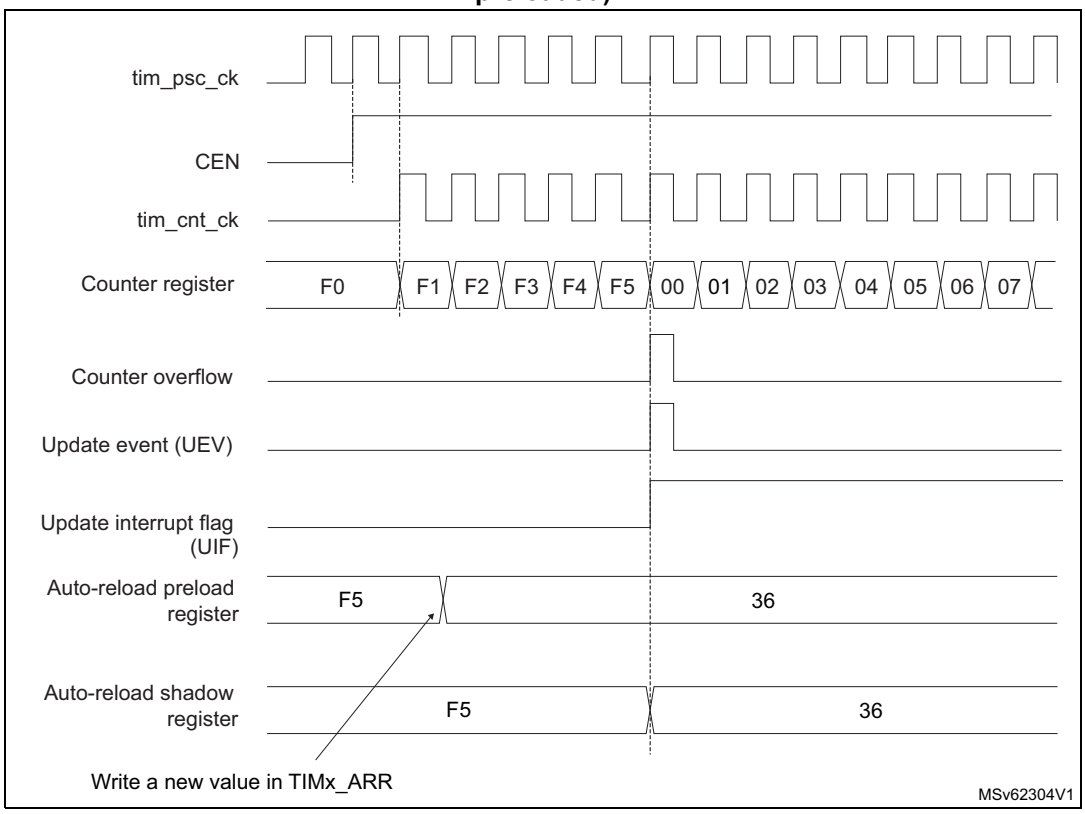


Figure 441. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 442. Counter timing diagram, internal clock divided by 1

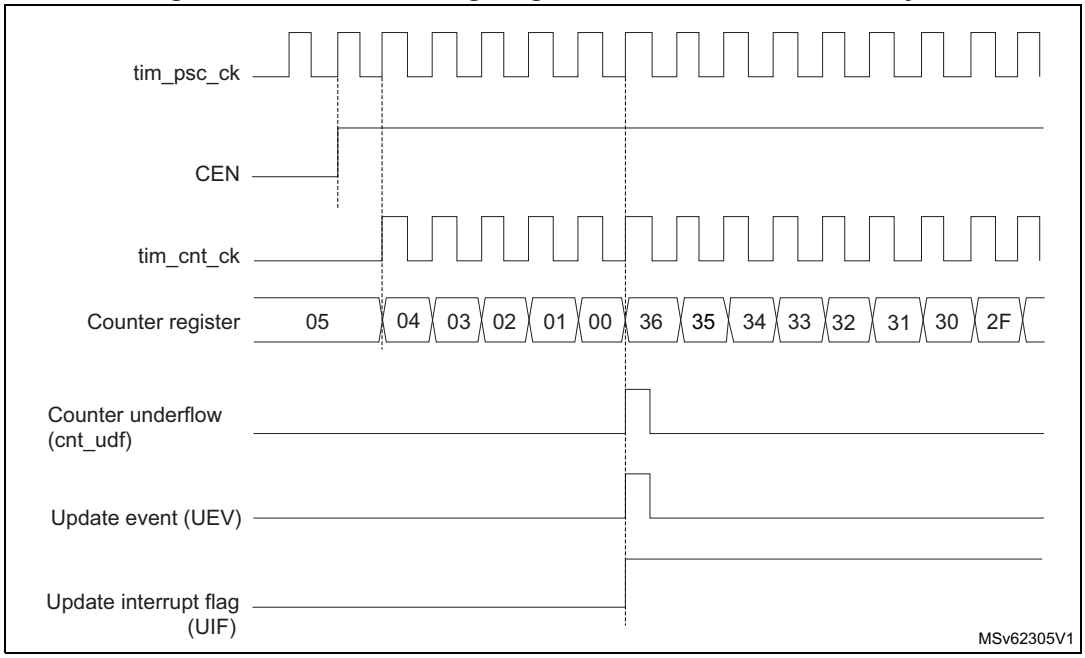


Figure 443. Counter timing diagram, internal clock divided by 2

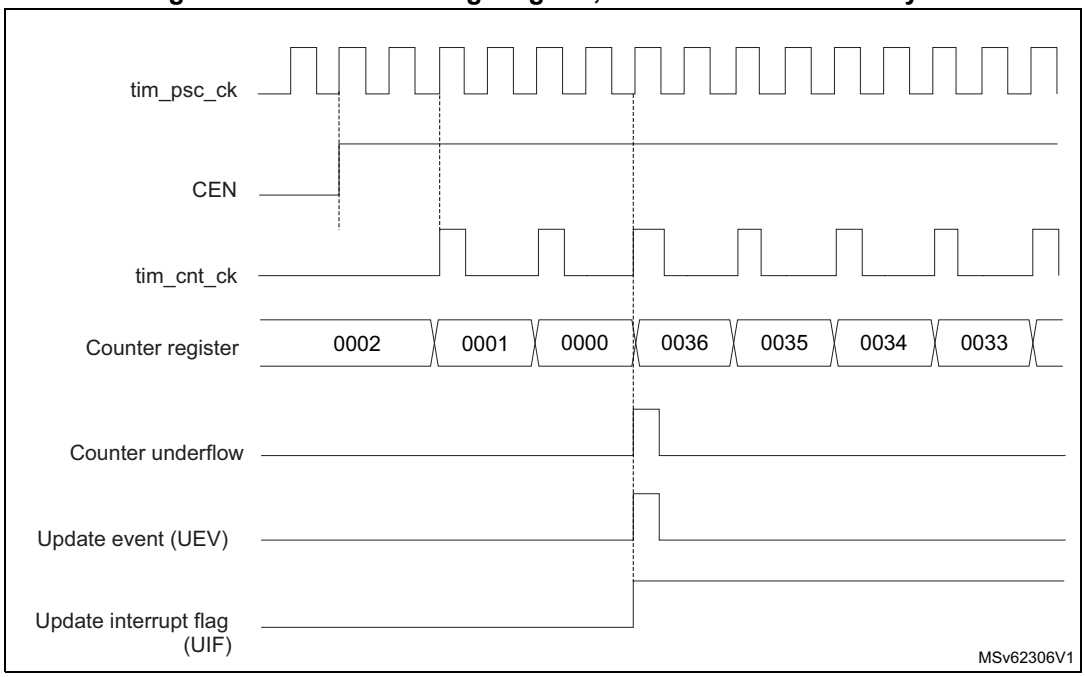


Figure 444. Counter timing diagram, internal clock divided by 4

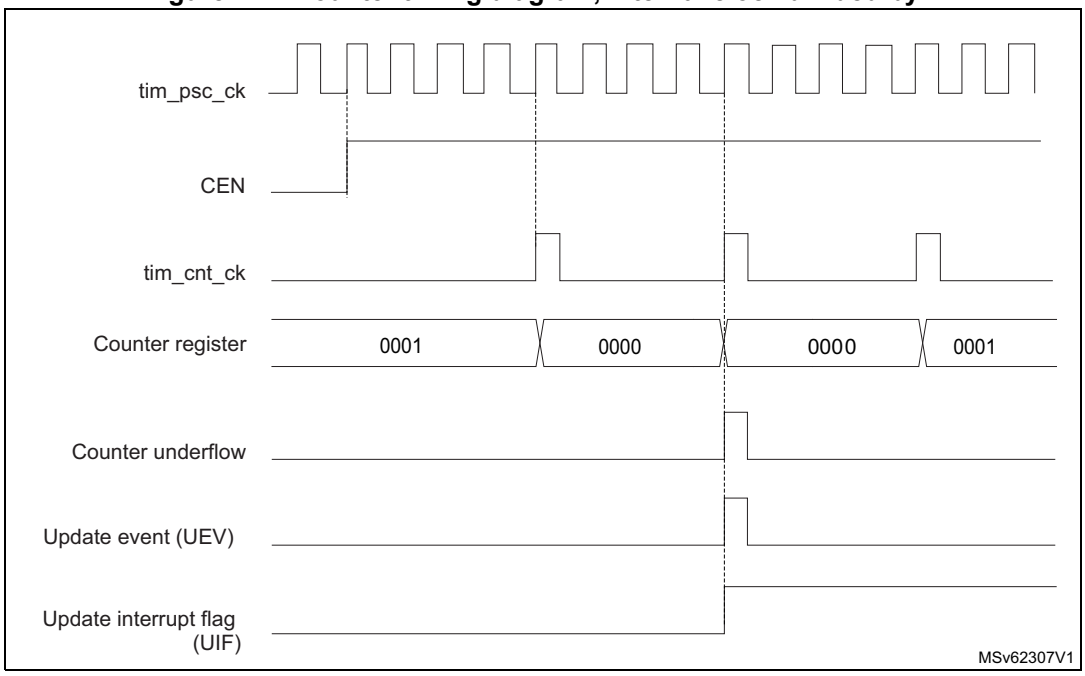


Figure 445. Counter timing diagram, internal clock divided by N

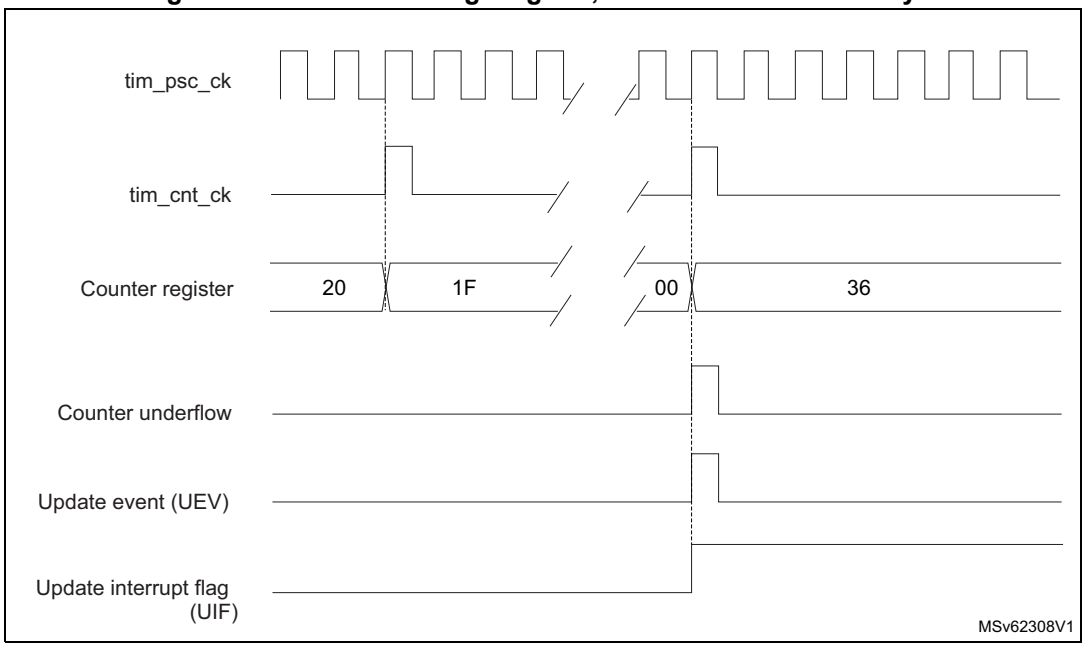
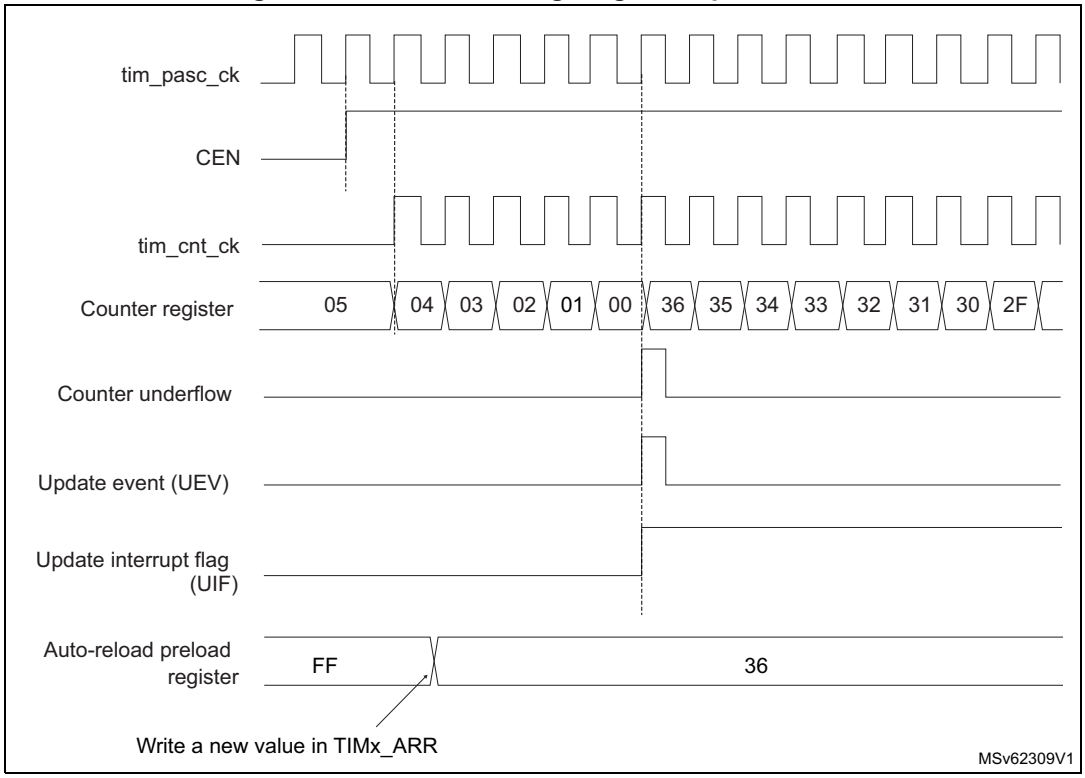


Figure 446. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

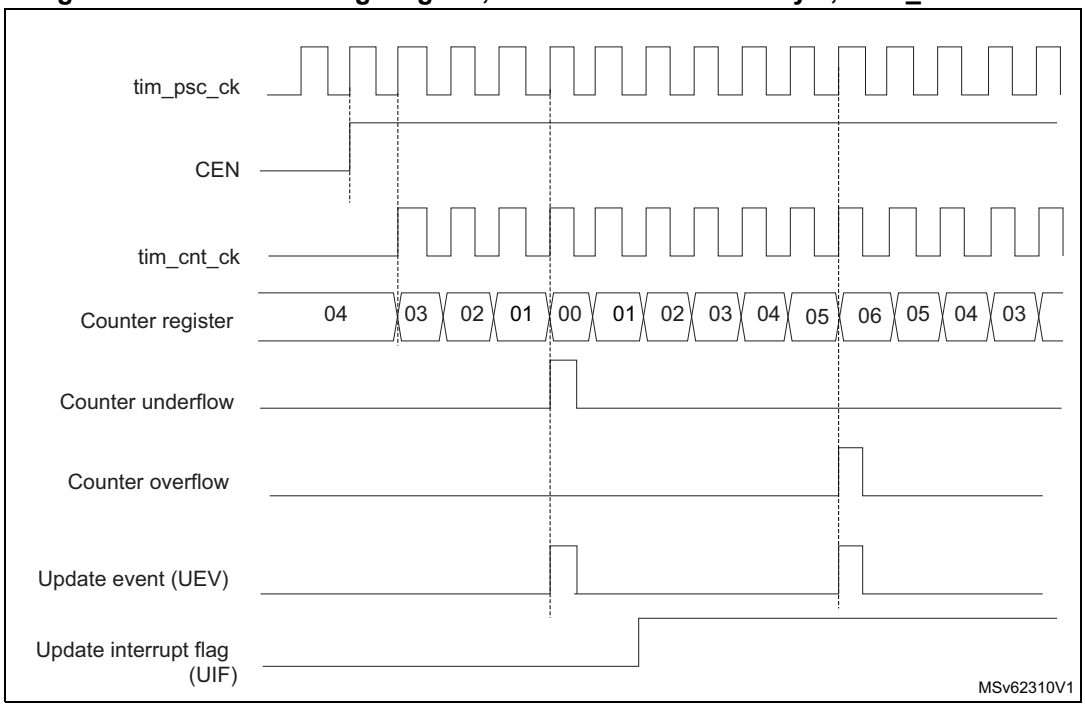
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 447. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 39.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5)*).

Figure 448. Counter timing diagram, internal clock divided by 2

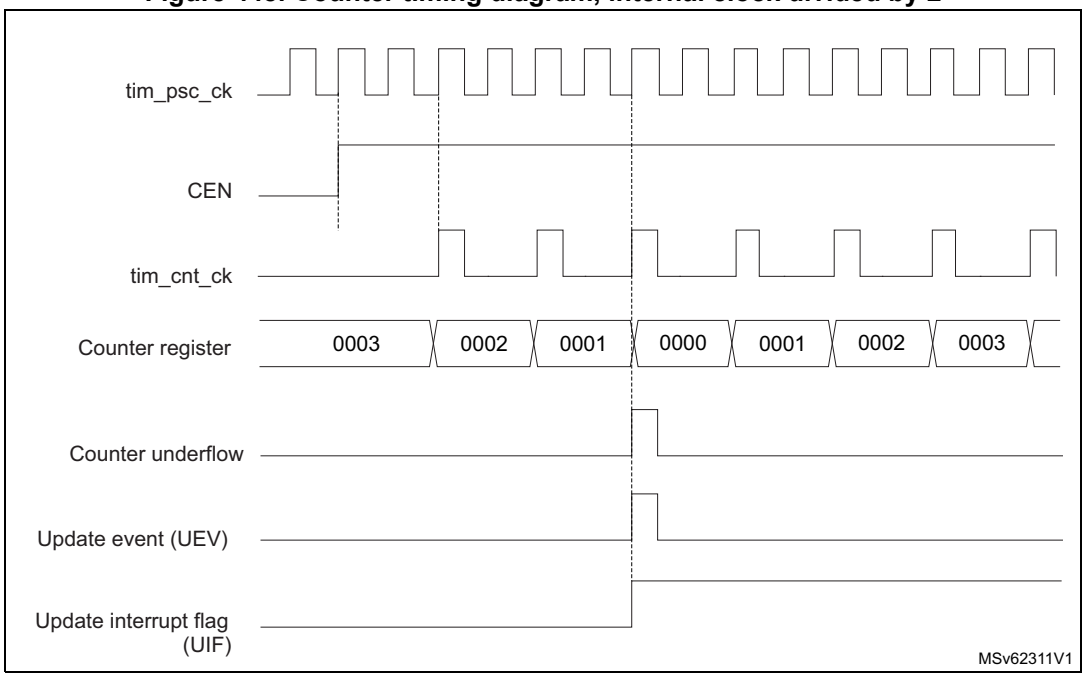
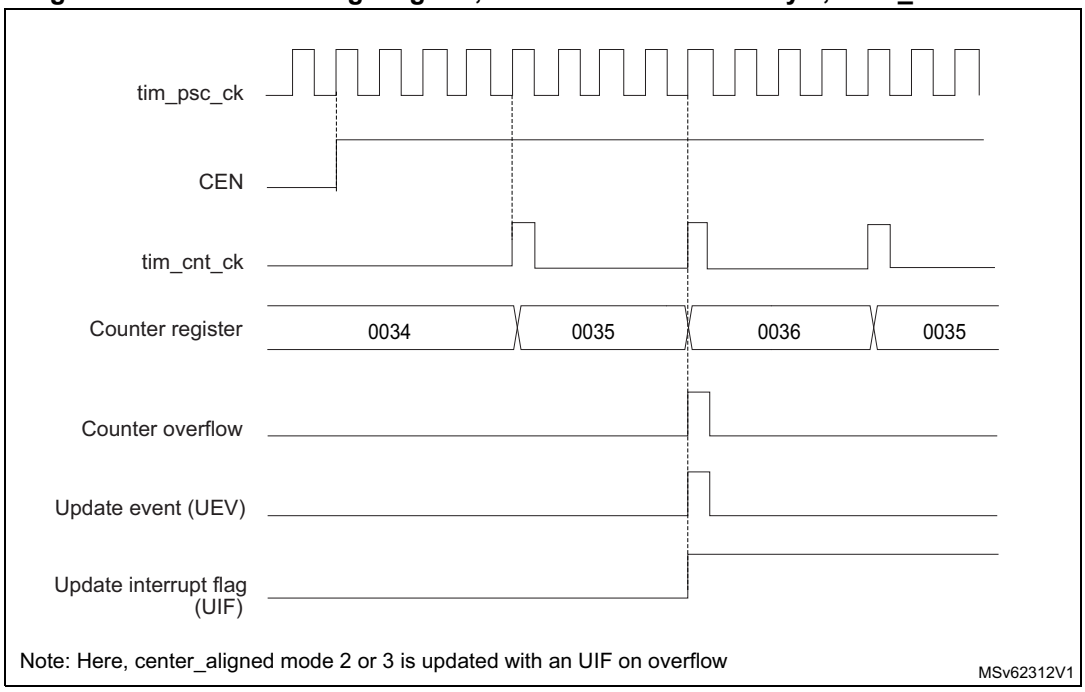


Figure 449. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 450. Counter timing diagram, internal clock divided by N

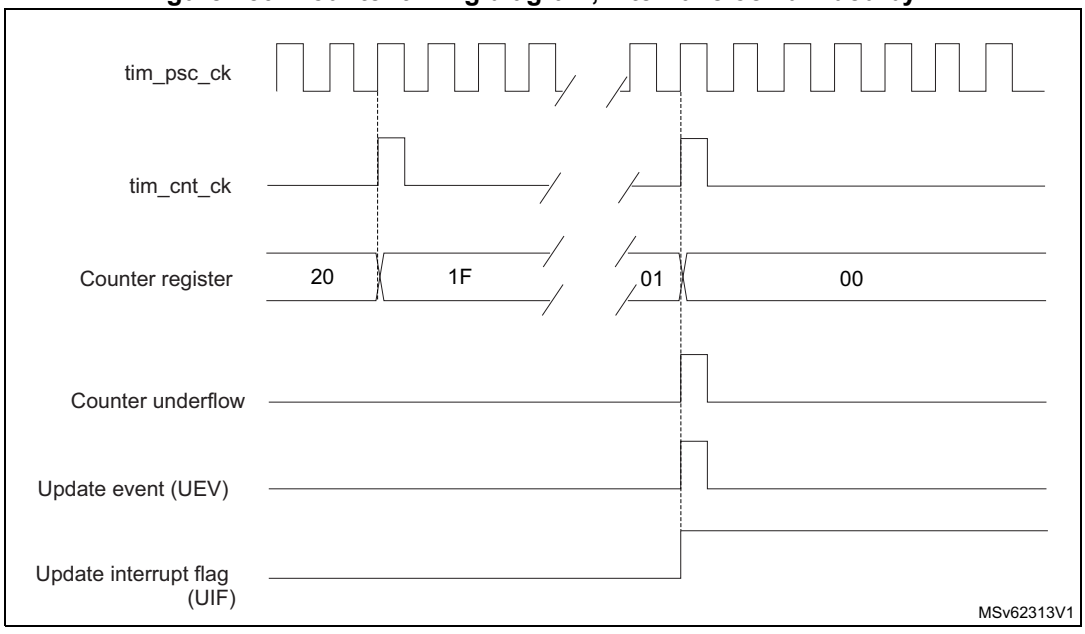


Figure 451. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

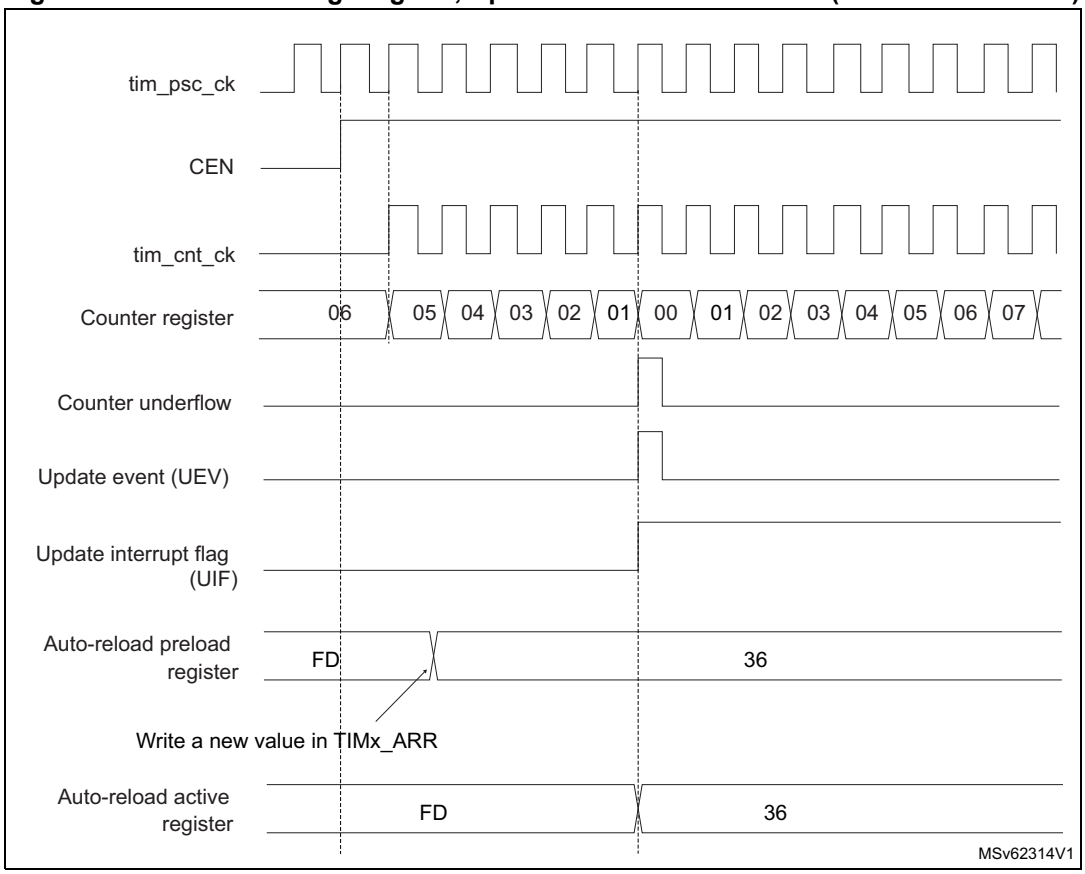
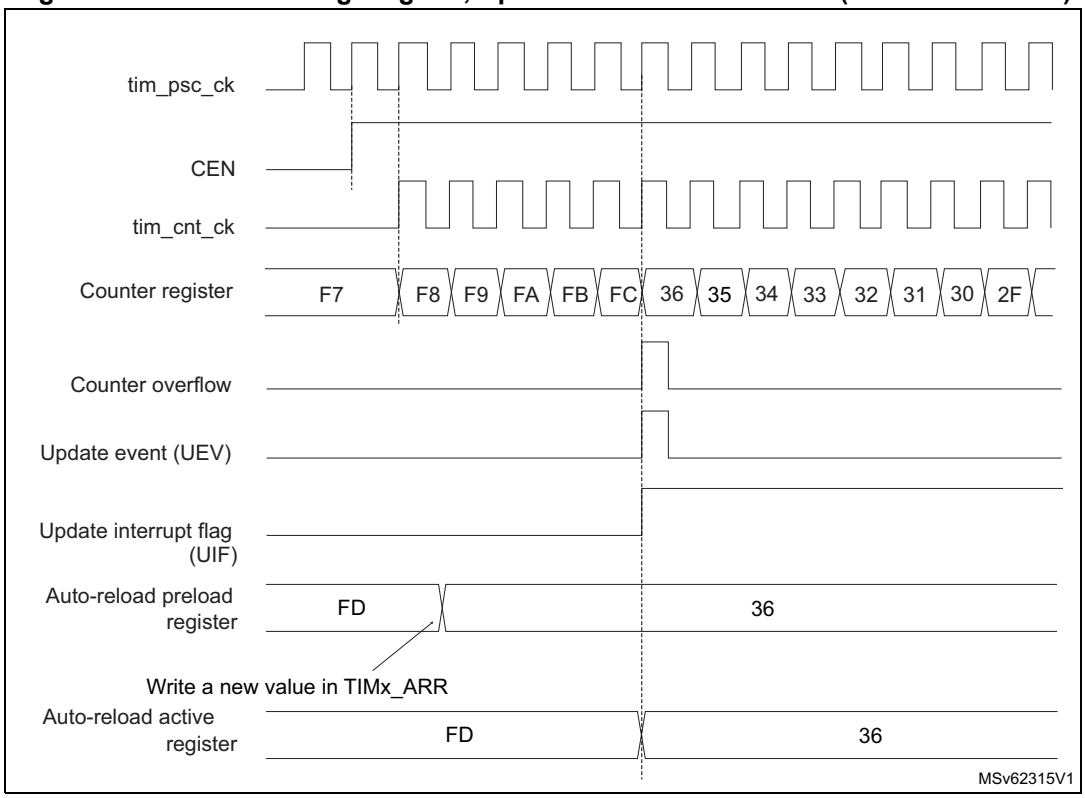


Figure 452. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



39.4.5 Clock selection

The counter clock can be provided by the following clock sources:

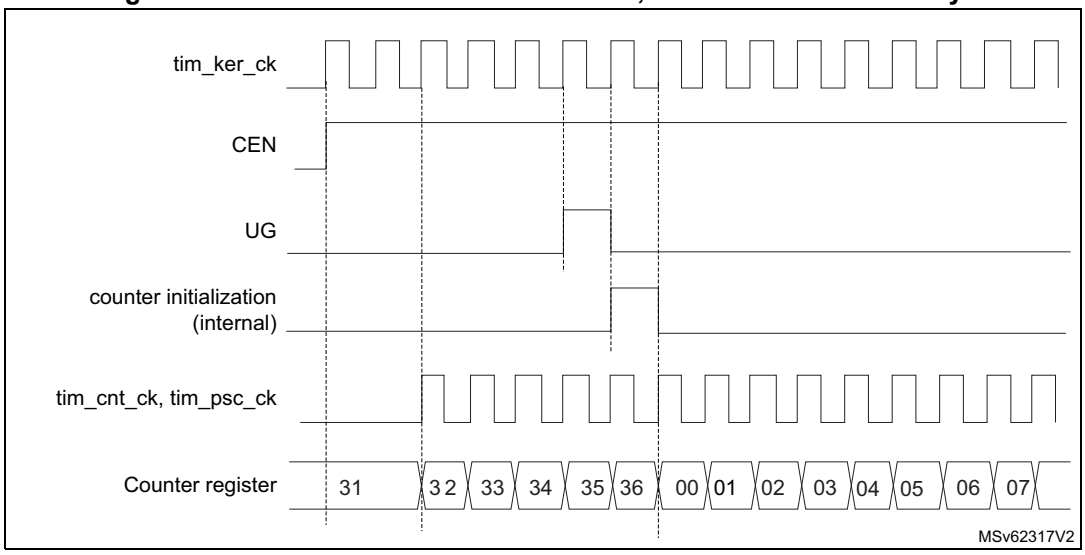
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register), and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 453 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

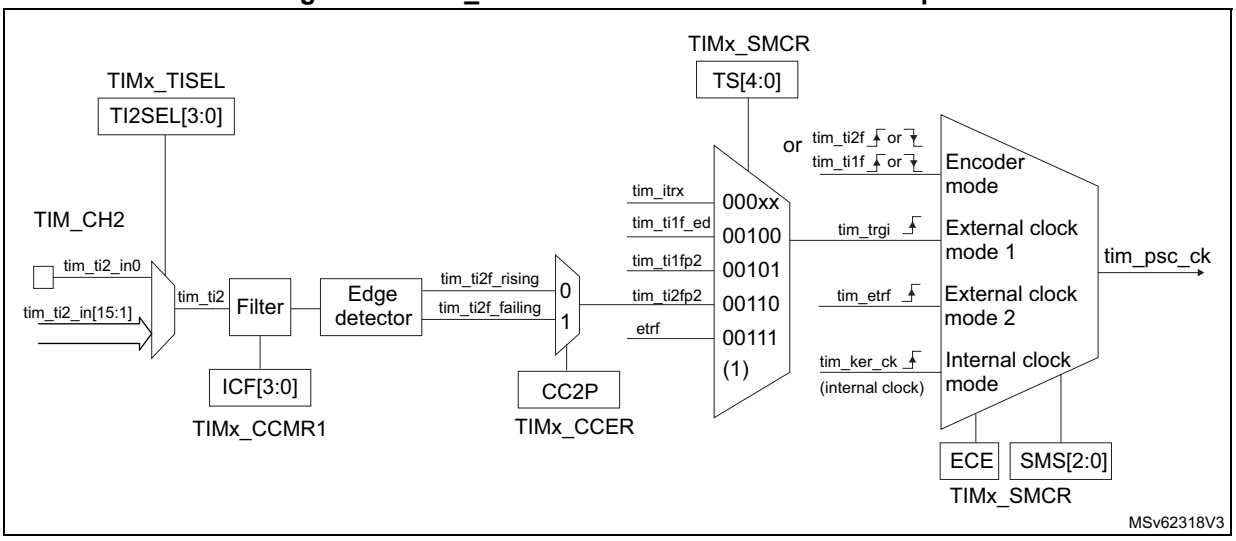
Figure 453. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 454. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111: tim_itr[15:0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).

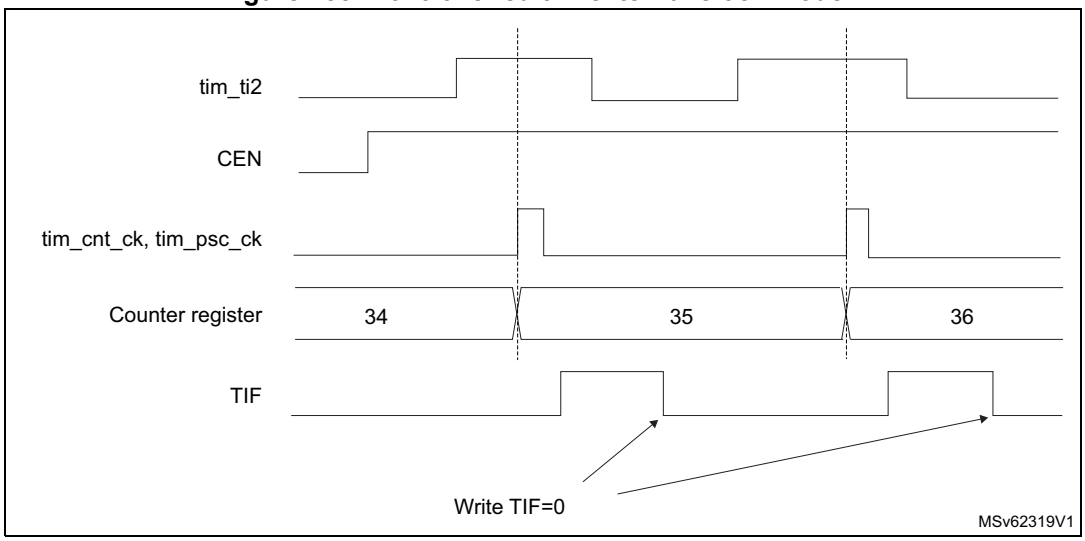
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 455. Control circuit in external clock mode 1



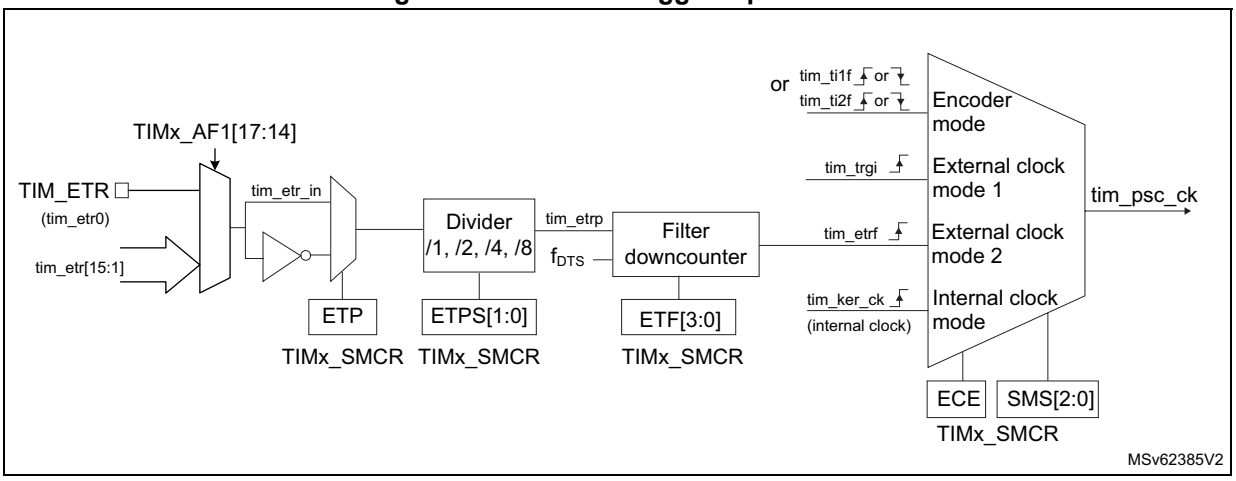
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 456 gives an overview of the external trigger input block.

Figure 456. External trigger input block



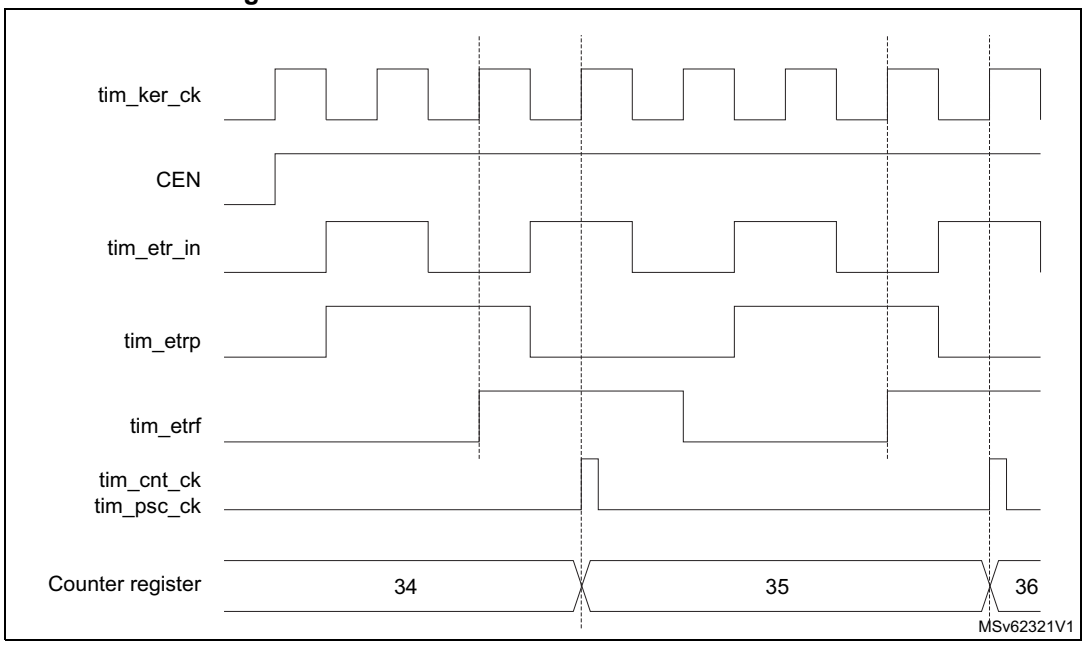
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 457. Control circuit in external clock mode 2



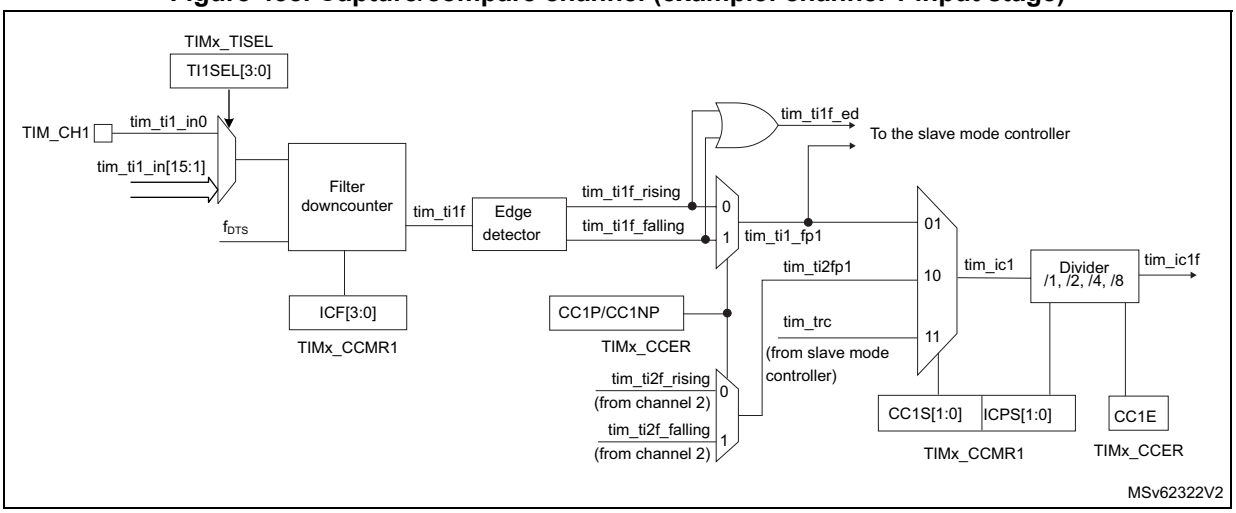
39.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 458. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 459. Capture/compare channel 1 main circuit

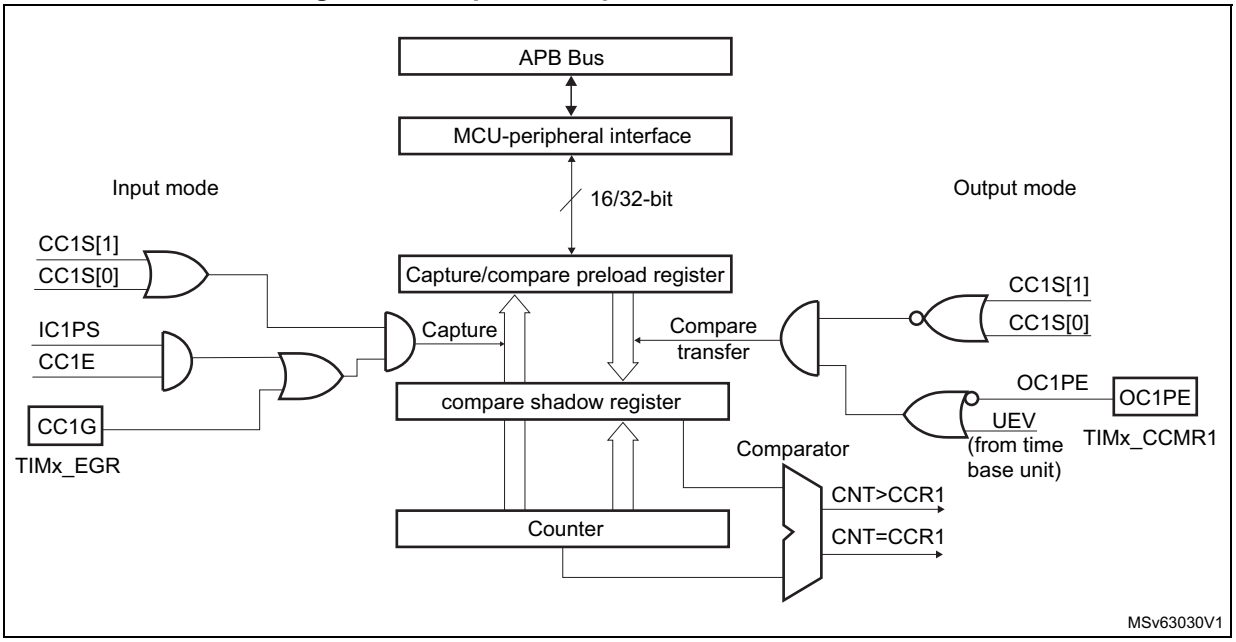
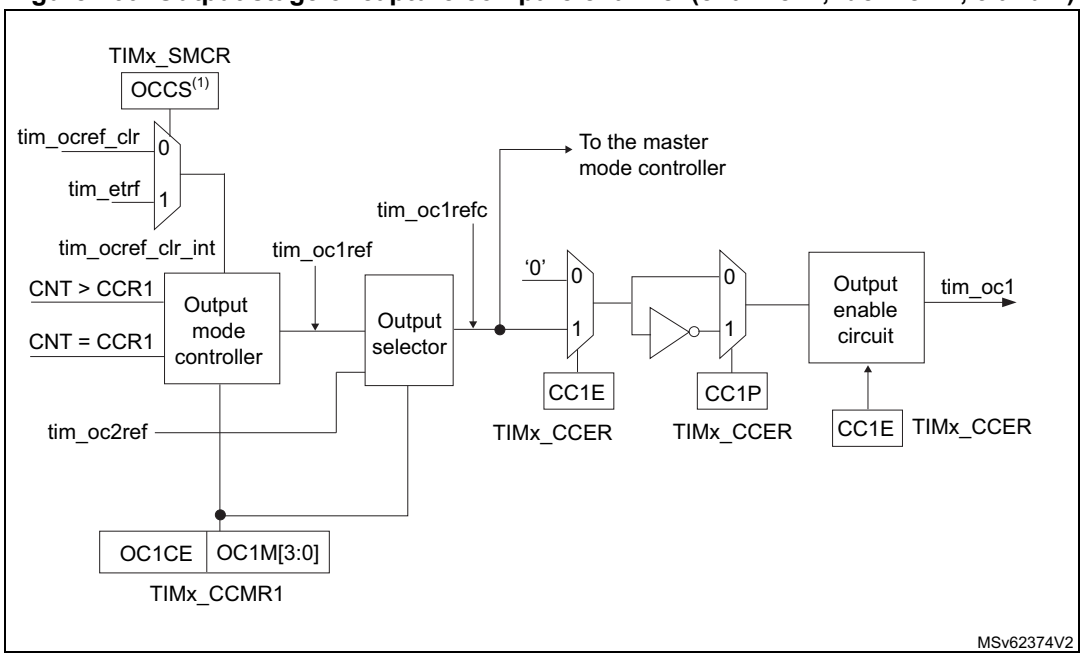


Figure 460. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

39.4.7 Input capture mode

In input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

39.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

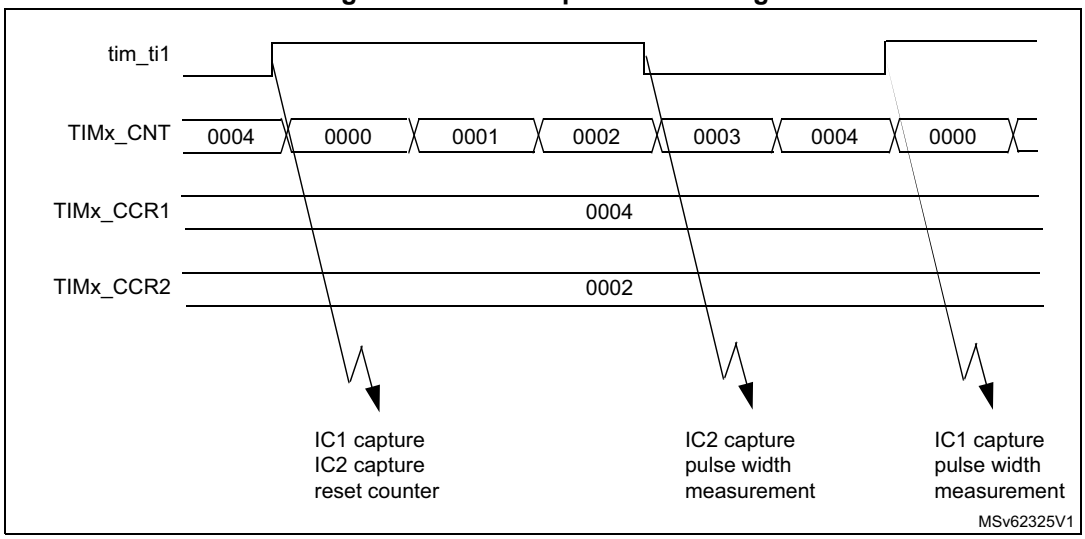
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 461. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

39.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare mode section.

39.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

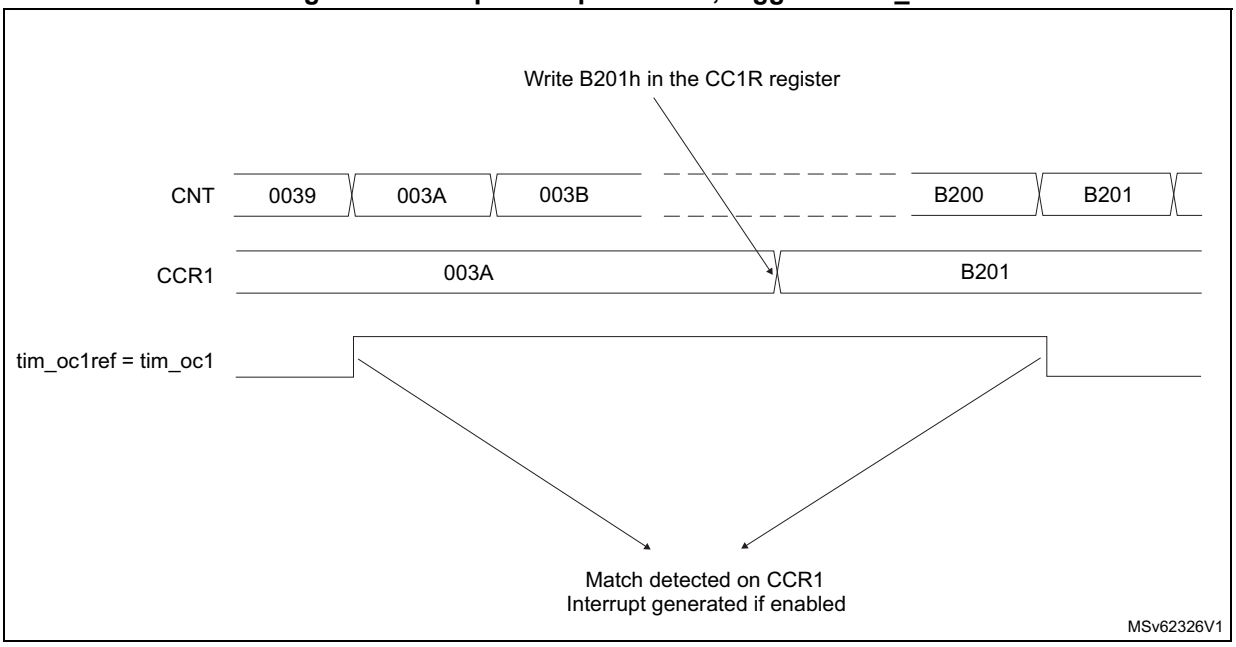
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 462*.

Figure 462. Output compare mode, toggle on tim_oc1



39.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 000) to one of the PWM modes (OCxM = 110 or 111). This forces the PWM by software while the timer is running.

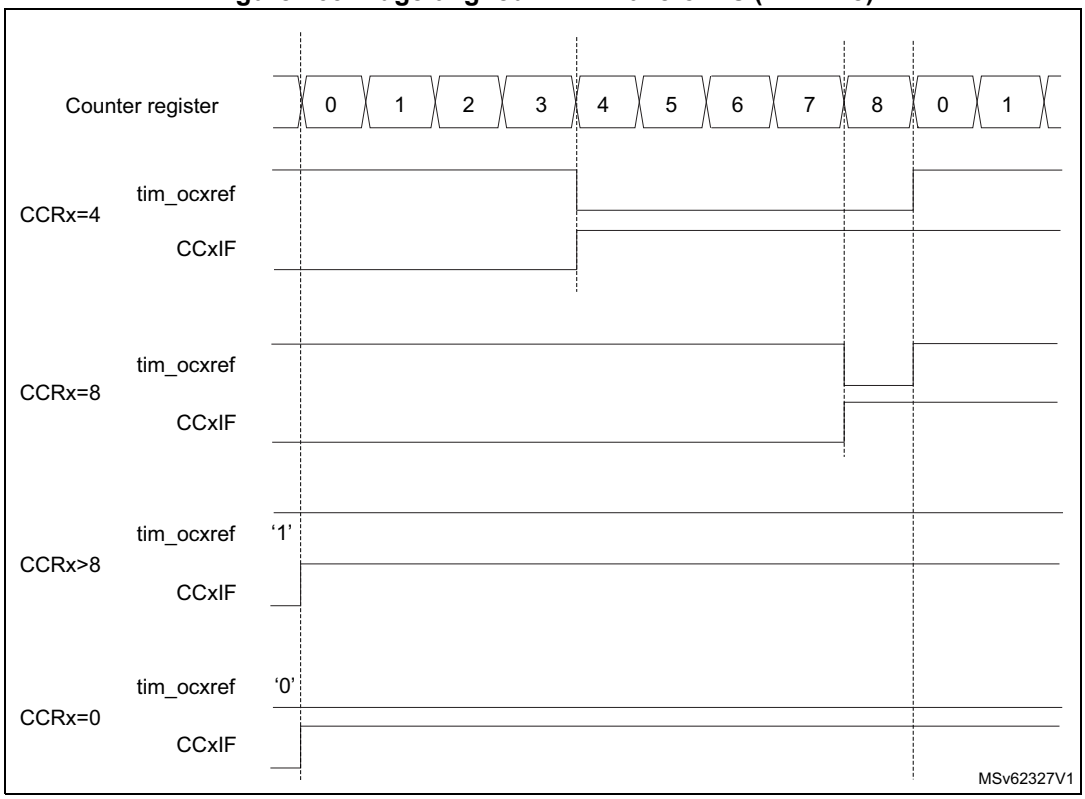
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 463* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 463. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIMx_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

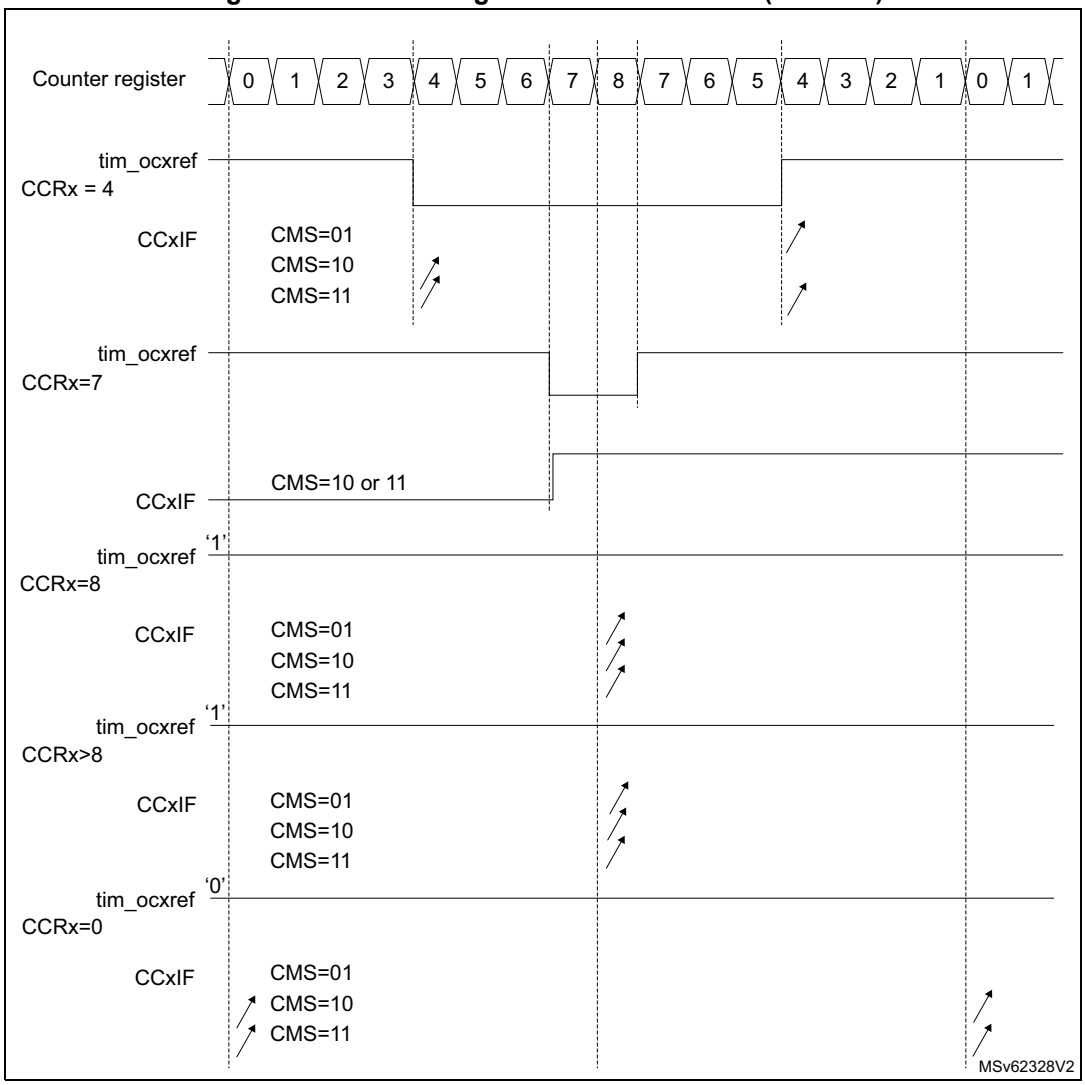
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 464 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 464. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

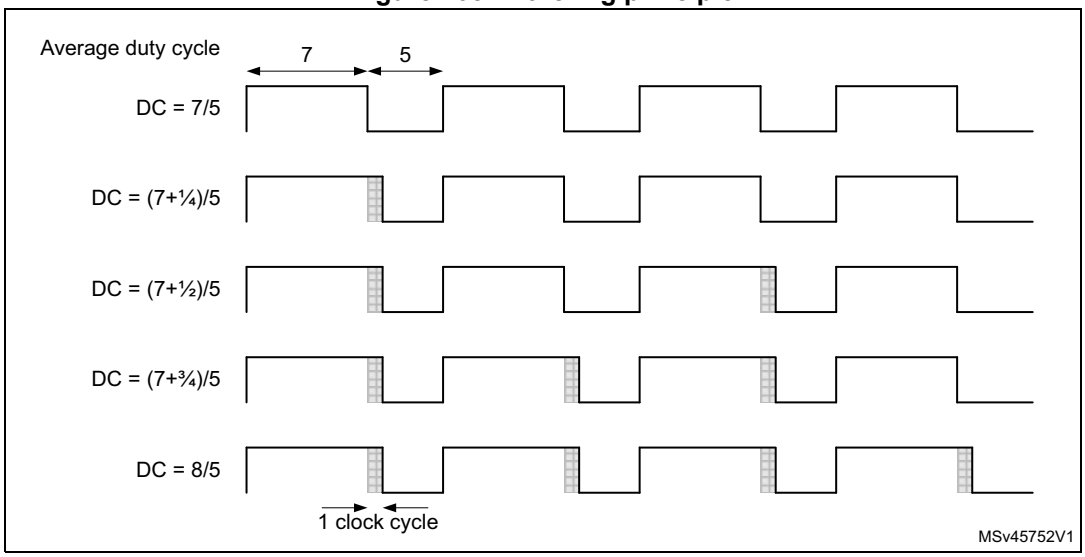
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 465* presents the dithering principle applied to four consecutive PWM cycles.

Figure 465. Dithering principle



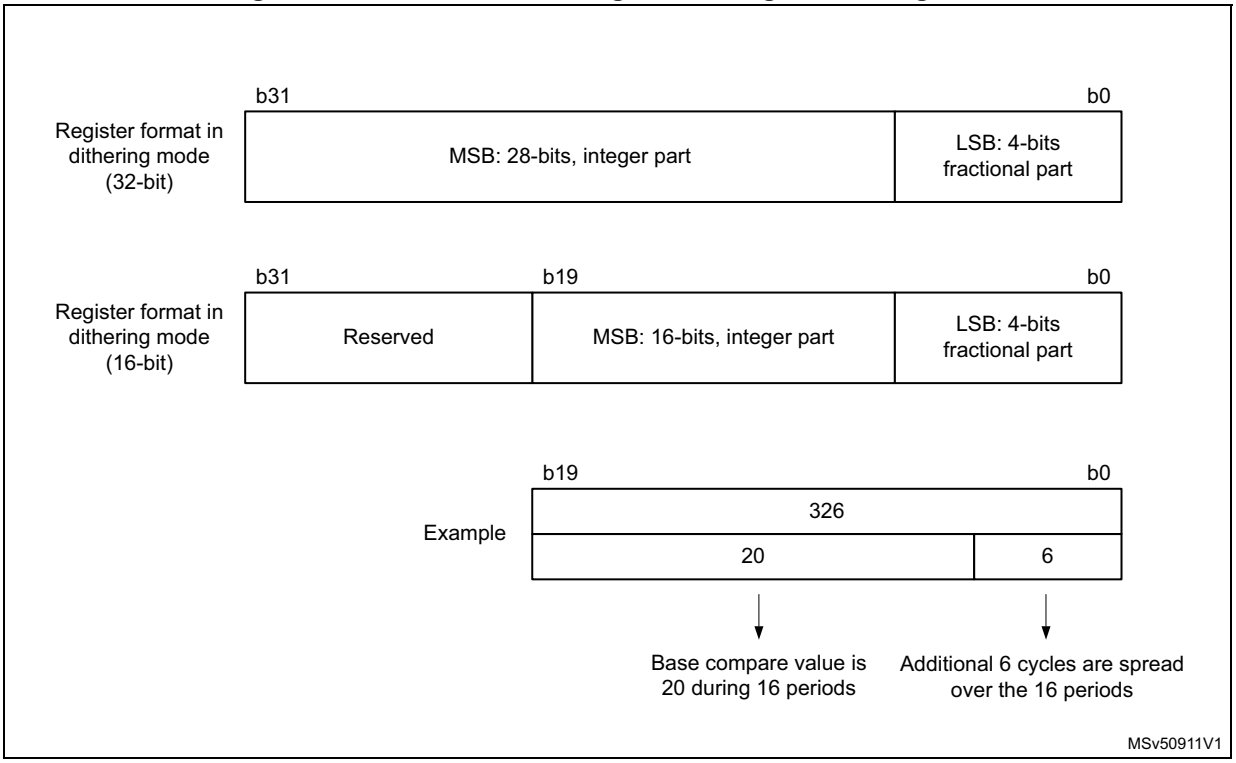
When the dithering mode is enabled, the register coding is changed as following (see *Figure 466* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 466. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 467* and *Figure 468*, the dithering mode is used to increase the PWM resolution.

Figure 467. PWM resolution vs frequency (16-bit mode)

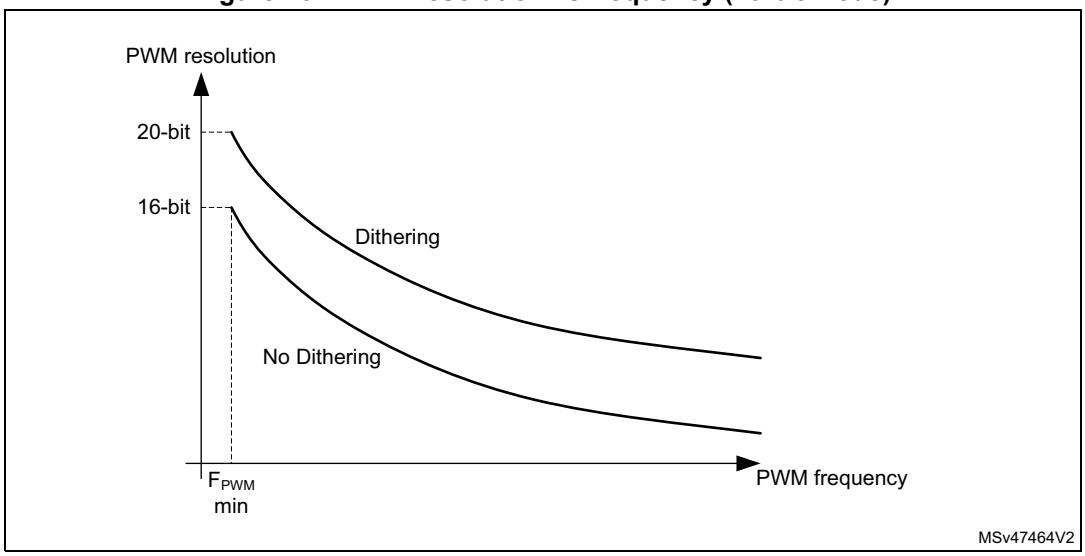
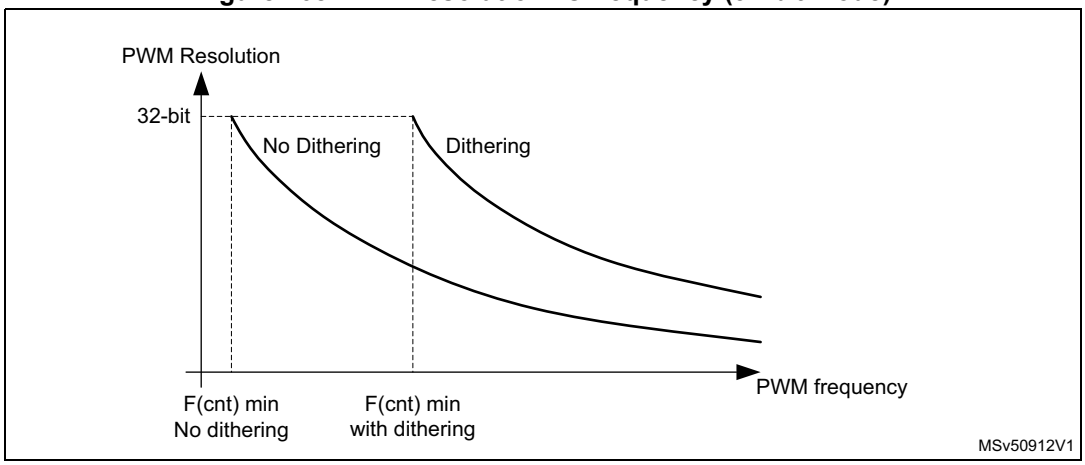
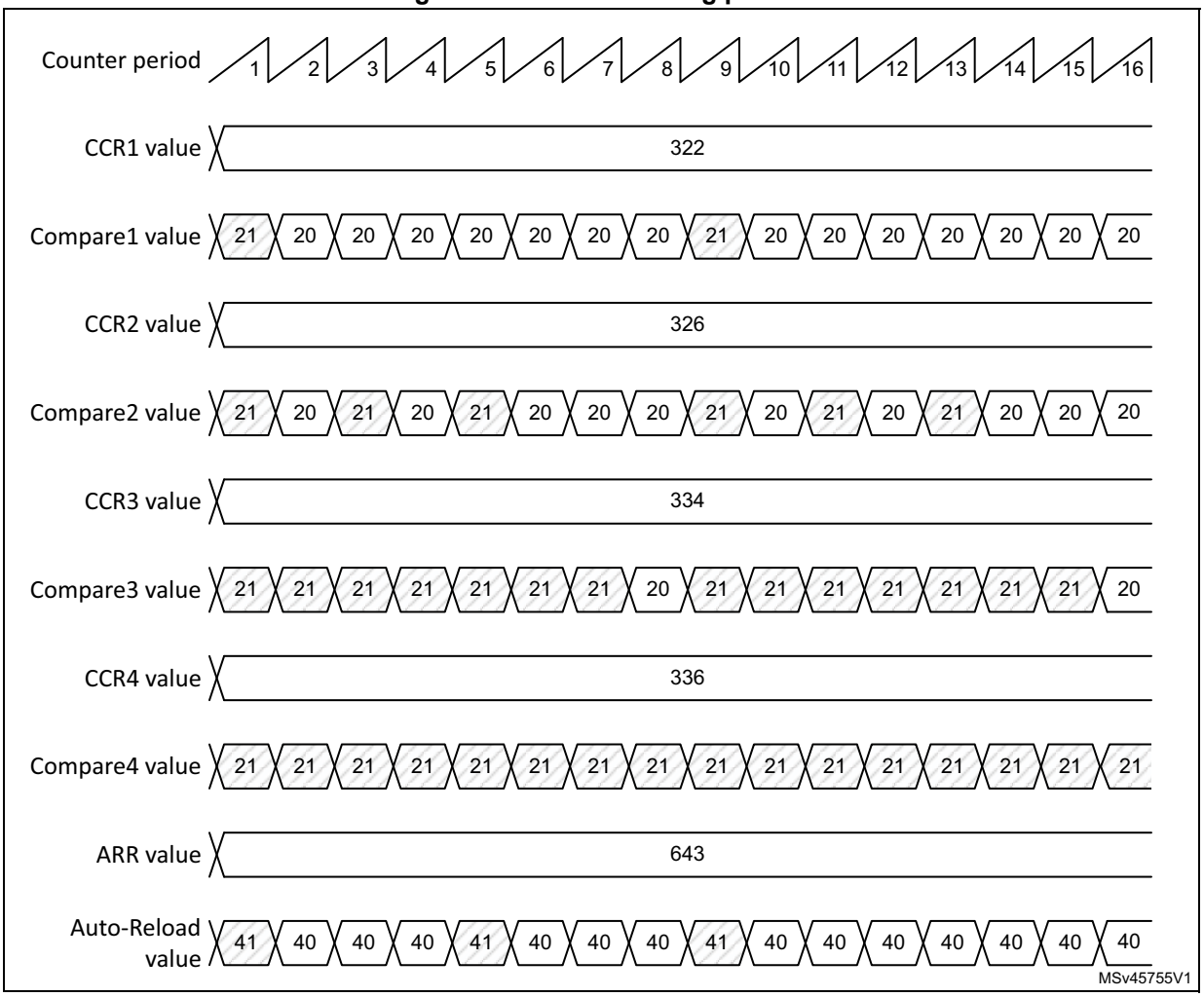


Figure 468. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 469*.

Figure 469. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 423*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 423. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 470*.

Figure 470. Dithering effect on duty cycle in center-aligned PWM mode

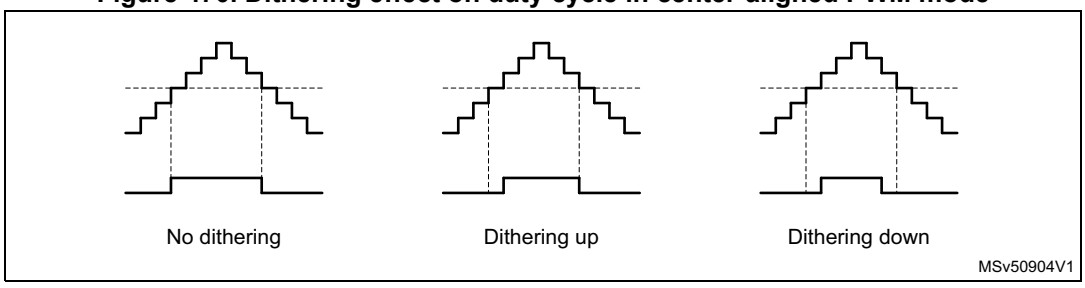


Table 424 shows how the dithering pattern is added in center-aligned PWM mode.

Table 424. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

39.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

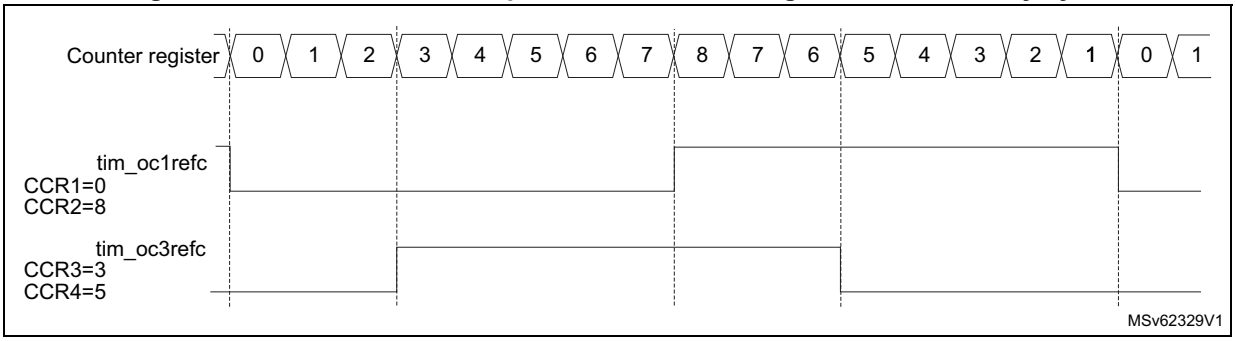
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 471 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 471. Generation of two phase-shifted PWM signals with 50% duty cycle



39.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

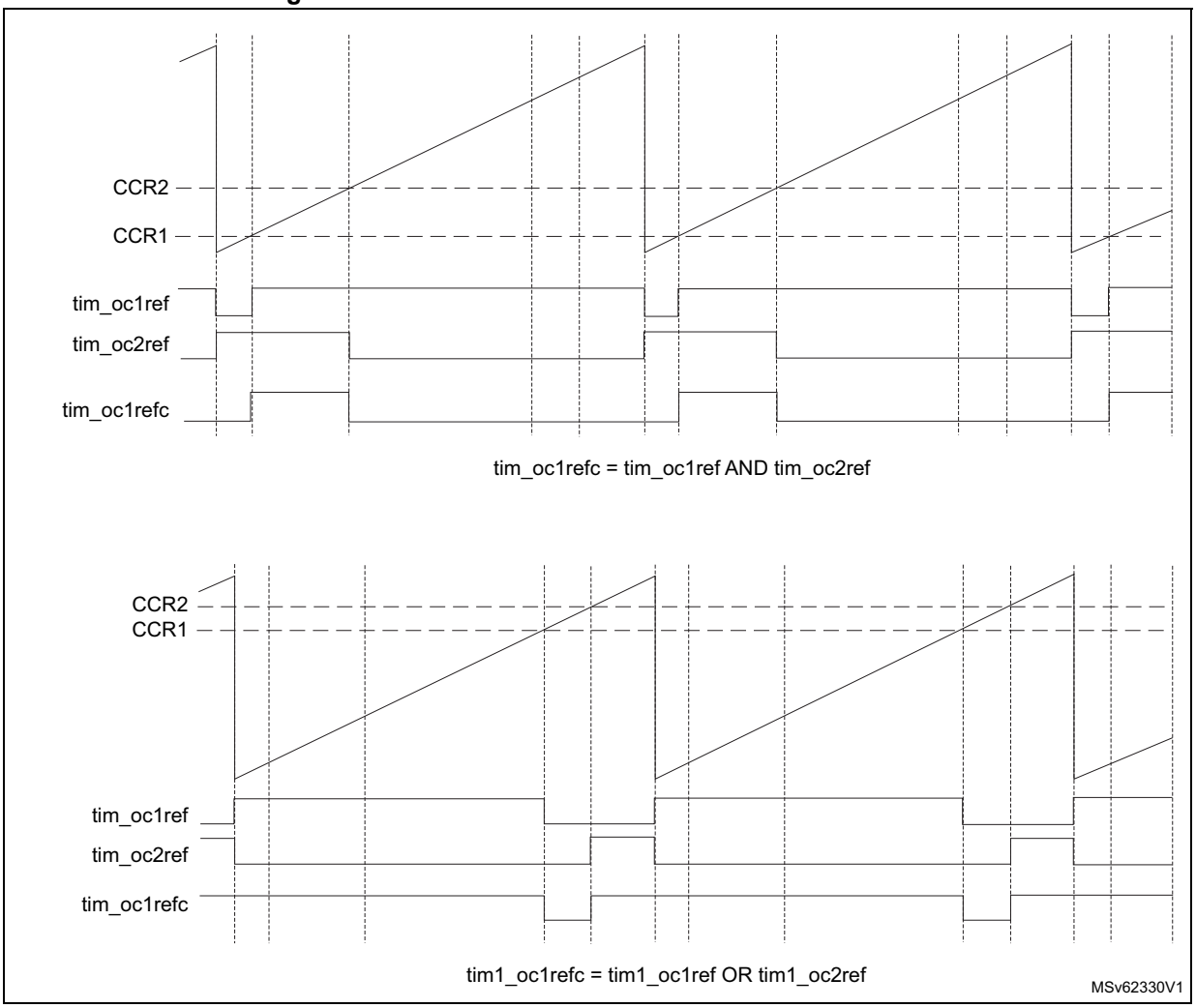
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 472 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 472. Combined PWM mode on channels 1 and 3



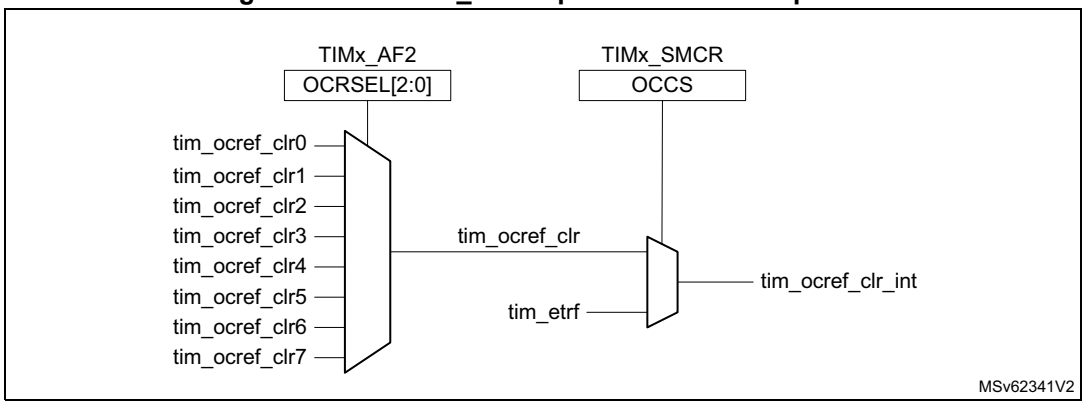
39.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 39.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in *Figure 473*.

Figure 473. OCREF_CLR input selection multiplexer



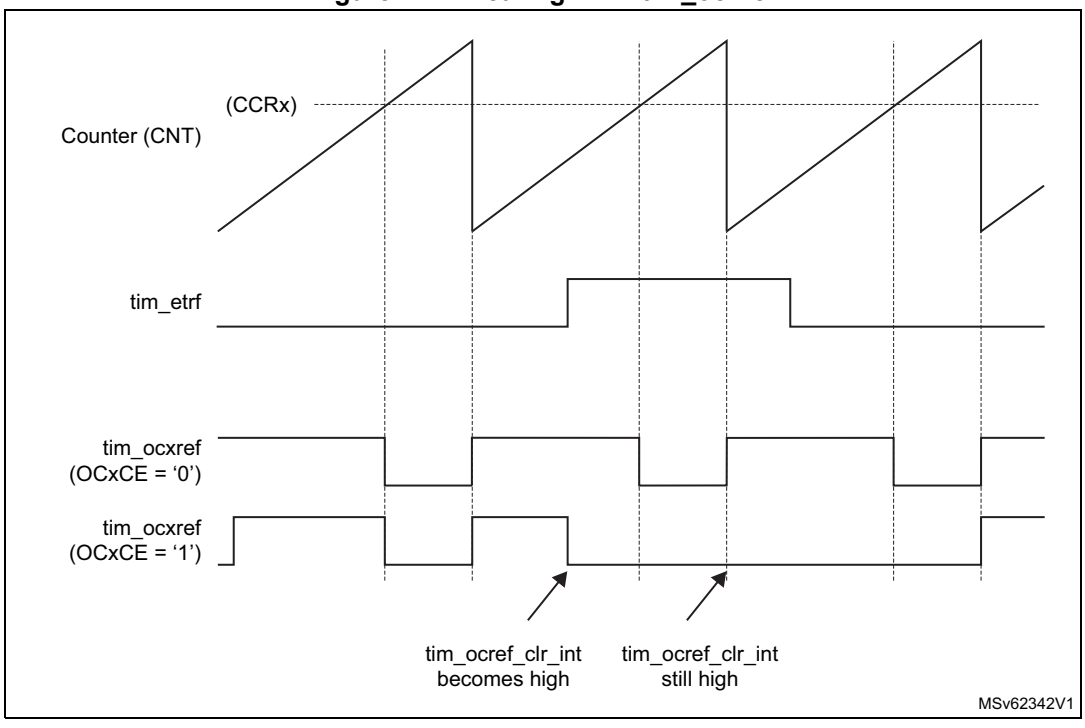
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 474 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 474. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

39.4.15 One-pulse mode

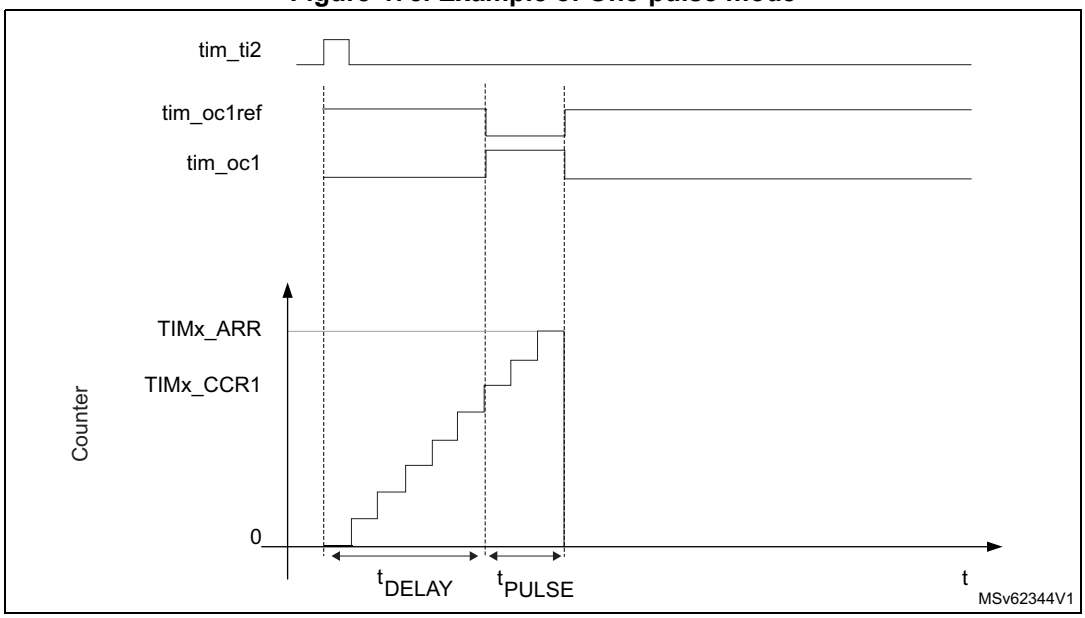
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 475. Example of One-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a one must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

39.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 39.4.15*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

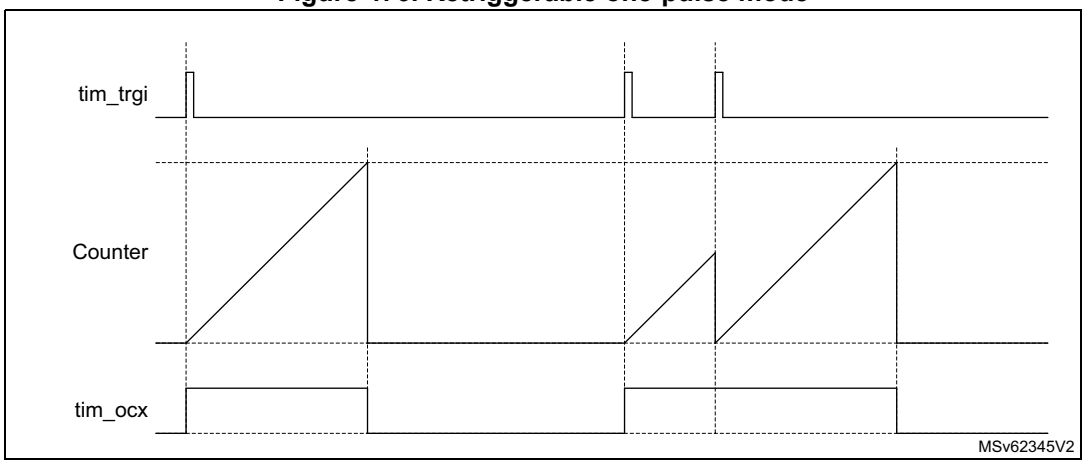
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 476. Retriggerable one-pulse mode



39.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 477*.

Figure 477. Pulse generator circuitry

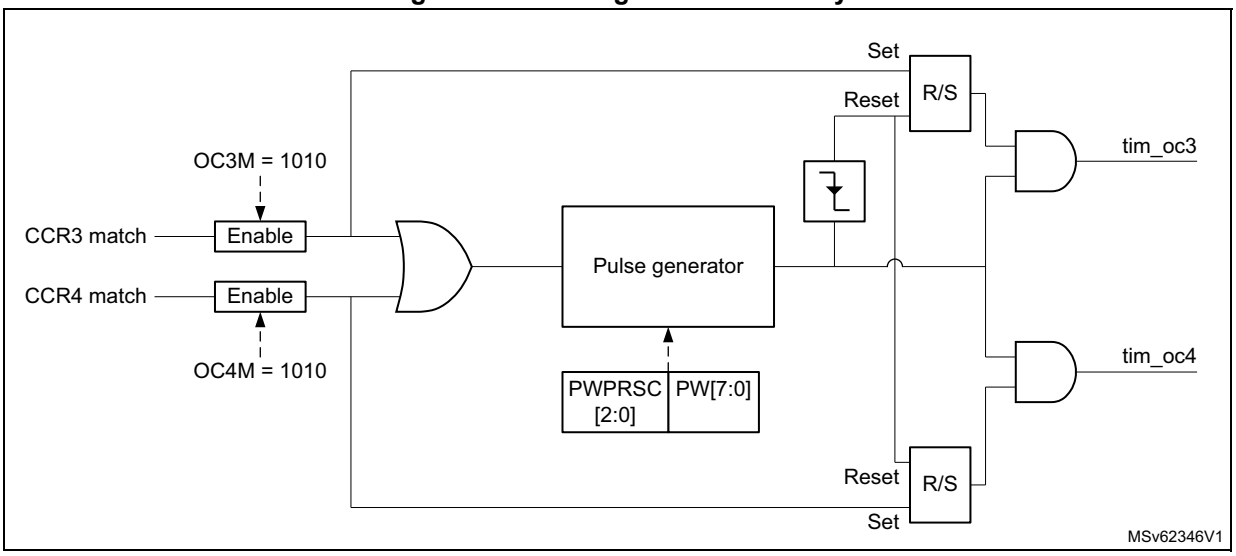
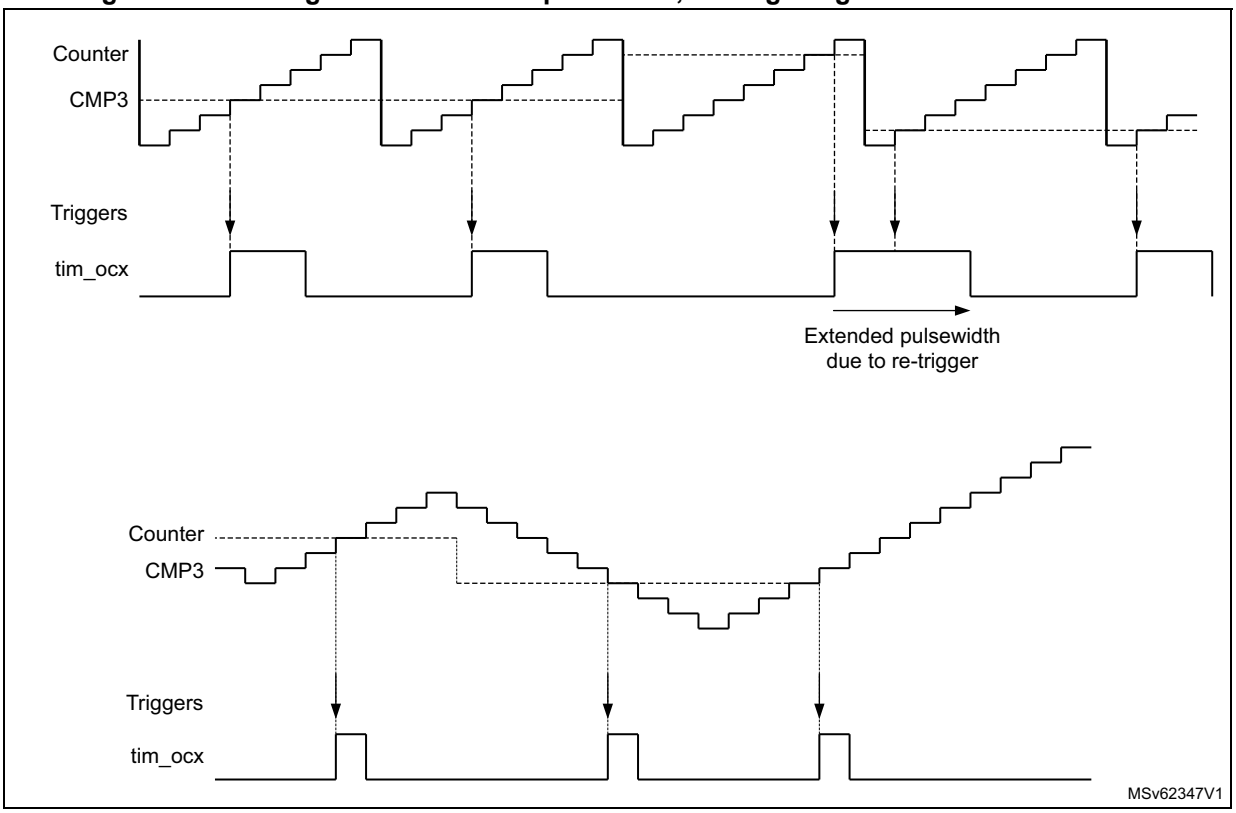


Figure 478 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 478. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

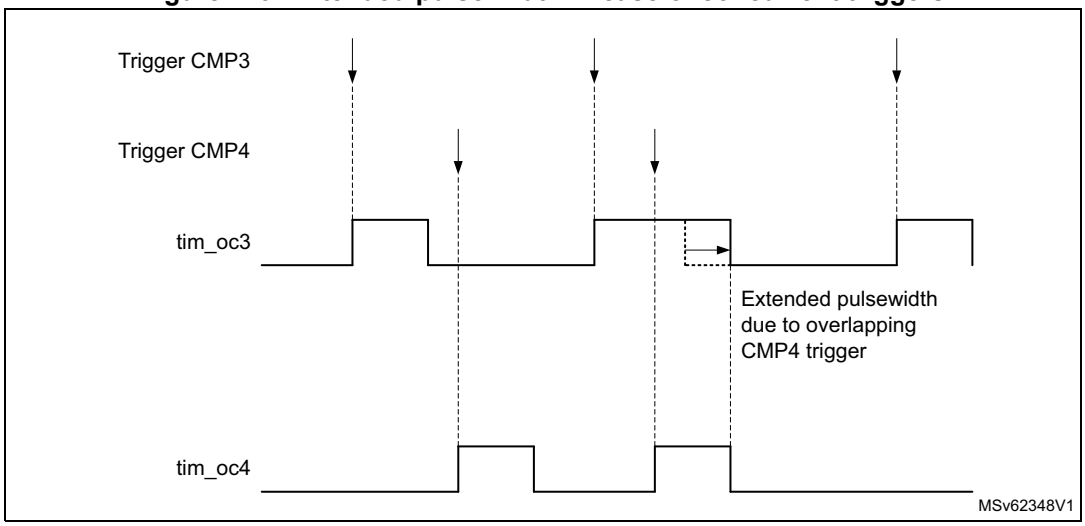
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 479).

Figure 479. Extended pulse width in case of concurrent triggers



39.4.18 Encoder interface mode

Quadrature encoder

To select Encoder interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 425*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or

tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 425. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 480 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are

selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, counter is enabled).

Figure 480. Example of counter operation in encoder interface mode

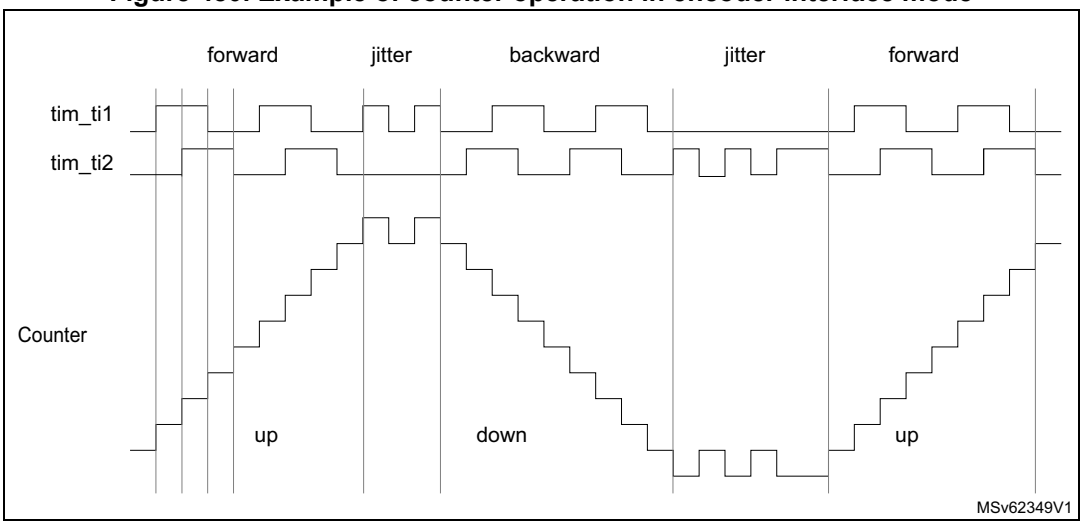


Figure 481 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 481. Example of encoder interface mode with tim_ti1fp1 polarity inverted

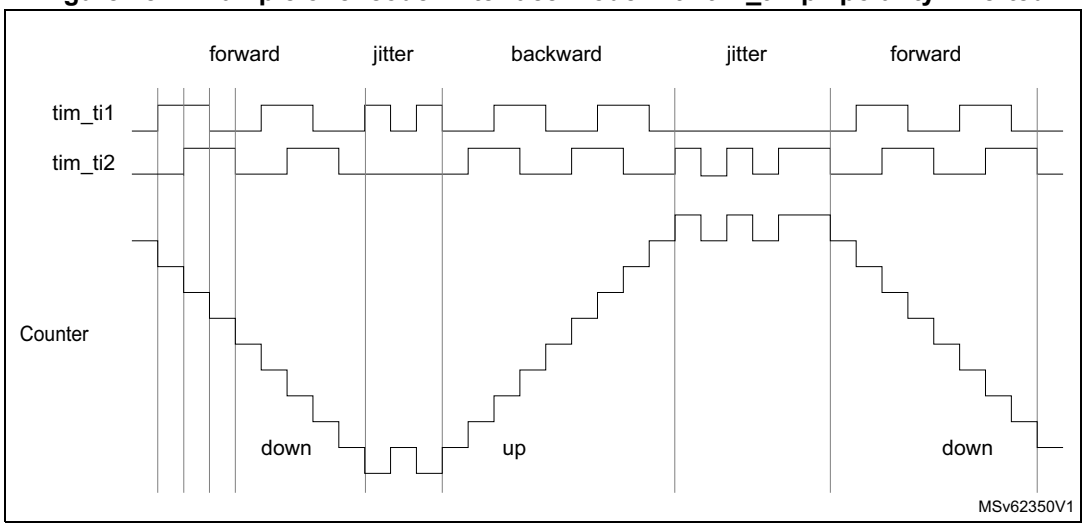
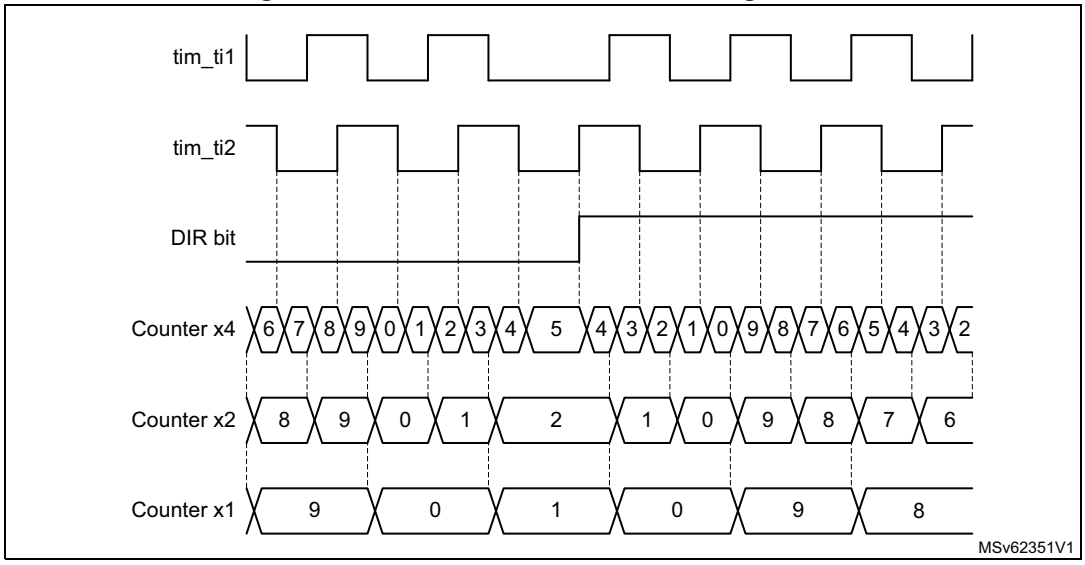


Figure 482 shows the timer counter value during a speed reversal, for various counting modes.

Figure 482. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

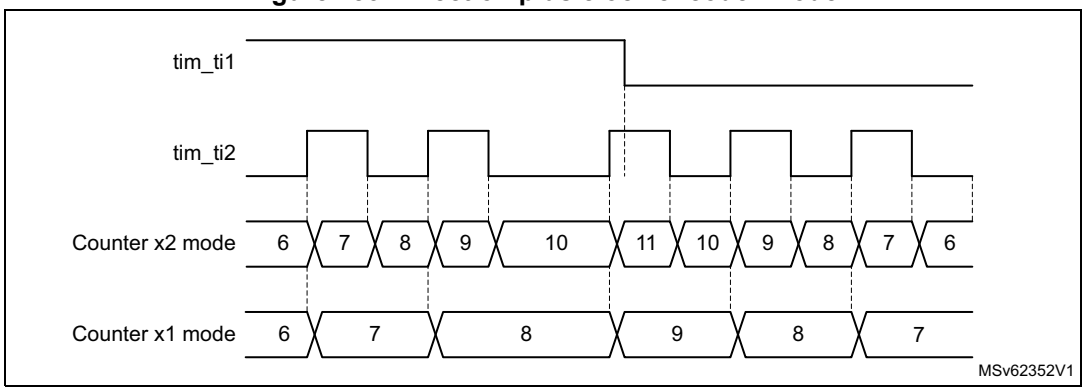
In the "clock plus direction" mode shown on *Figure 483*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 483. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 484*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 484*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 485*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 484*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 485*).

Figure 484. Directional clock encoder mode (CC1P = CC2P = 0)

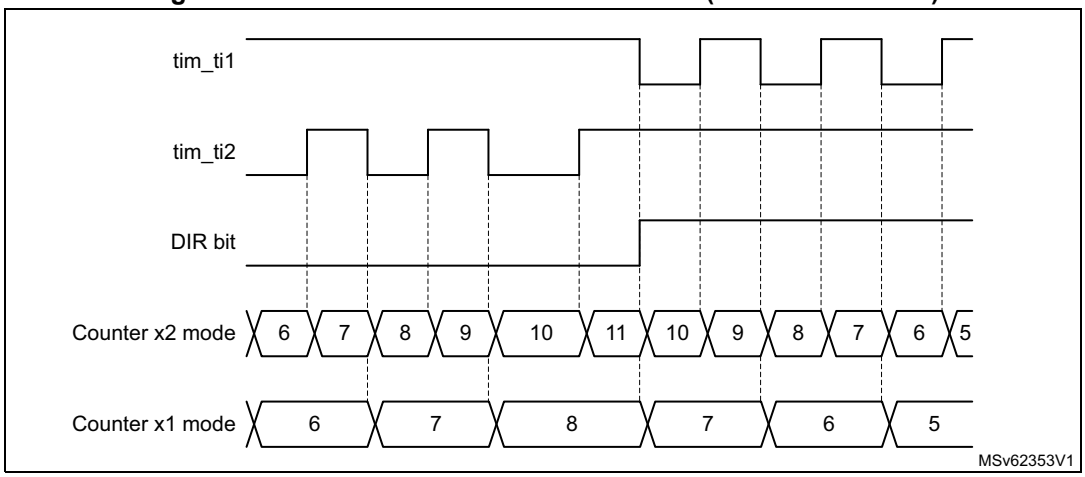


Figure 485. Directional clock encoder mode (CC1P = CC2P = 1)

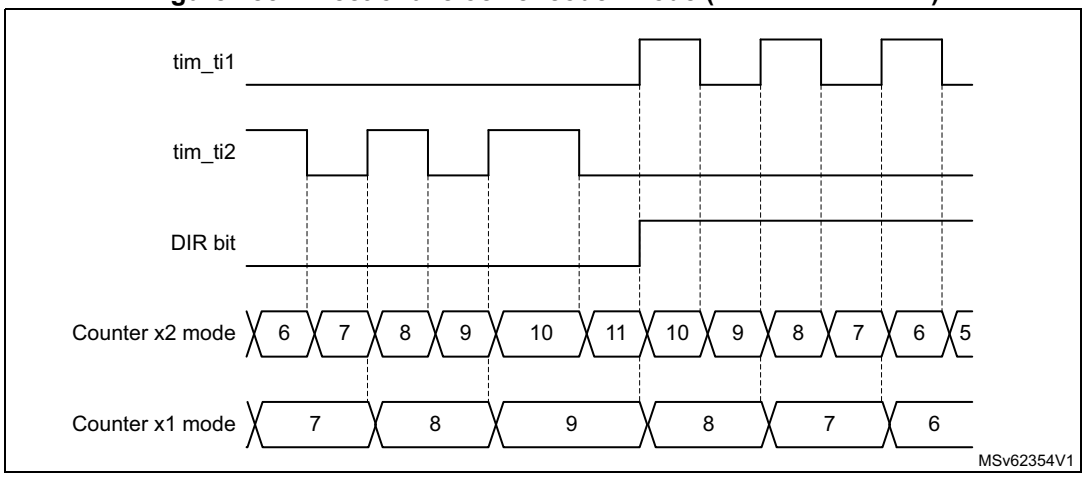


Table 426 details how the directional clock mode operates, for any input transition.

Table 426. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

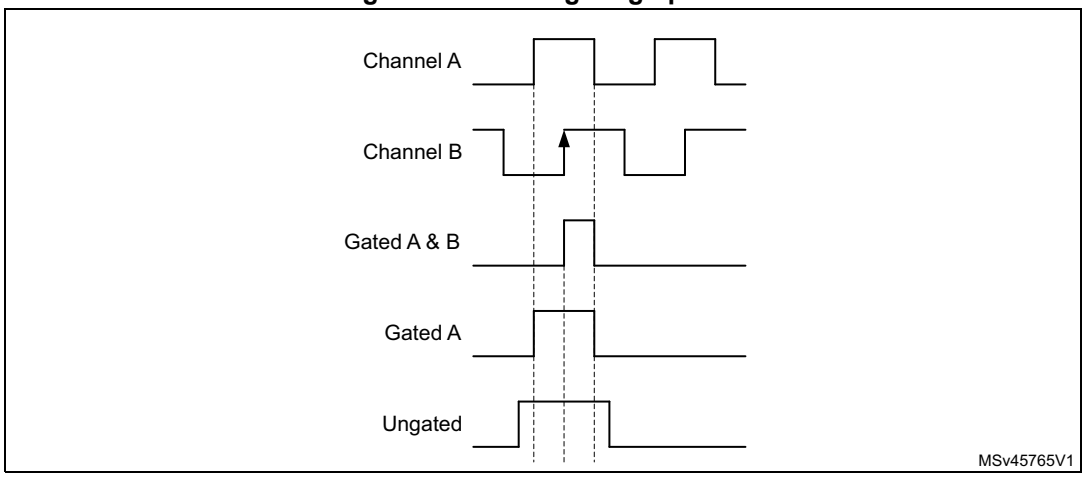
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 486*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

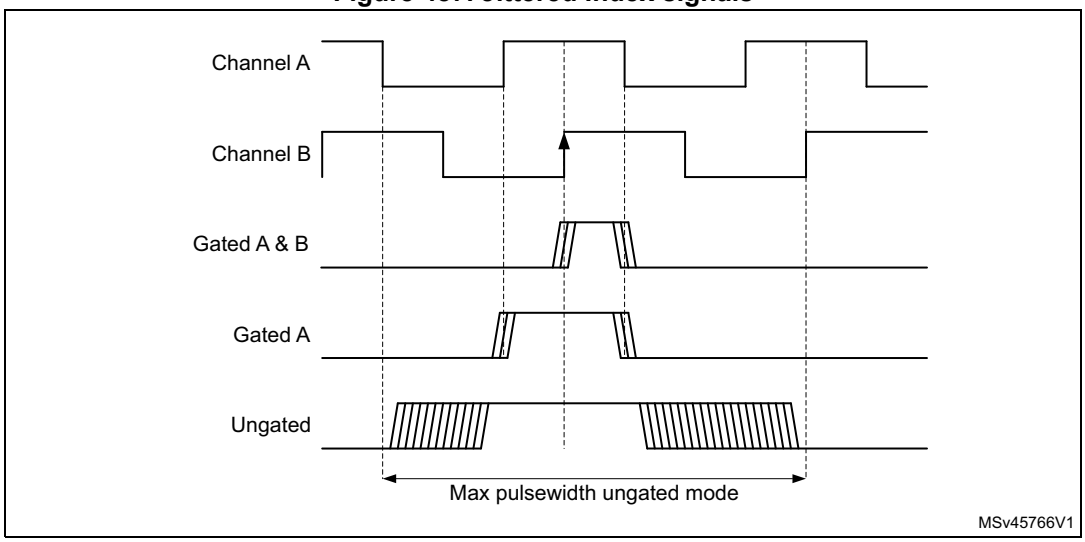
Figure 486. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 487*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 487. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 488* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 488. Index generation for IPOS[1:0] = 11

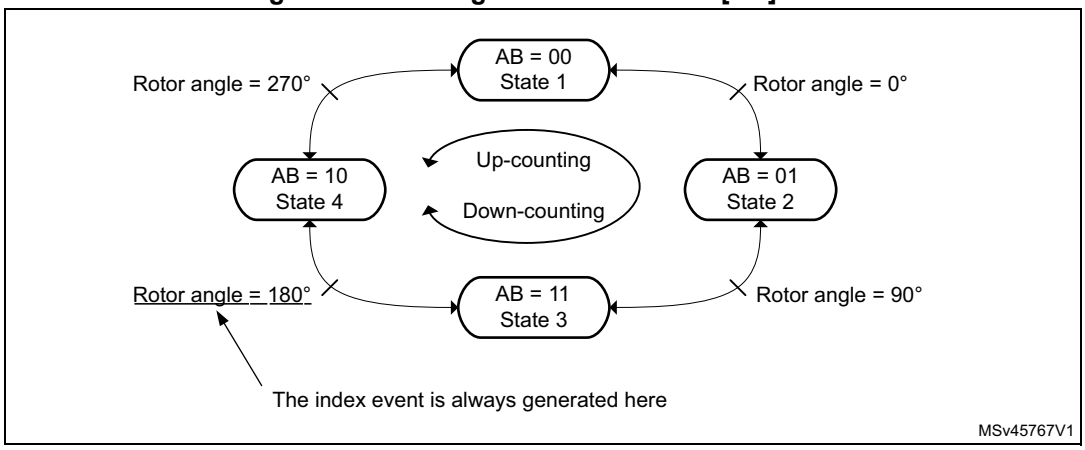


Figure 489 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 489. Counter reading with index gated on channel A (IPOS[1:0] = 11)

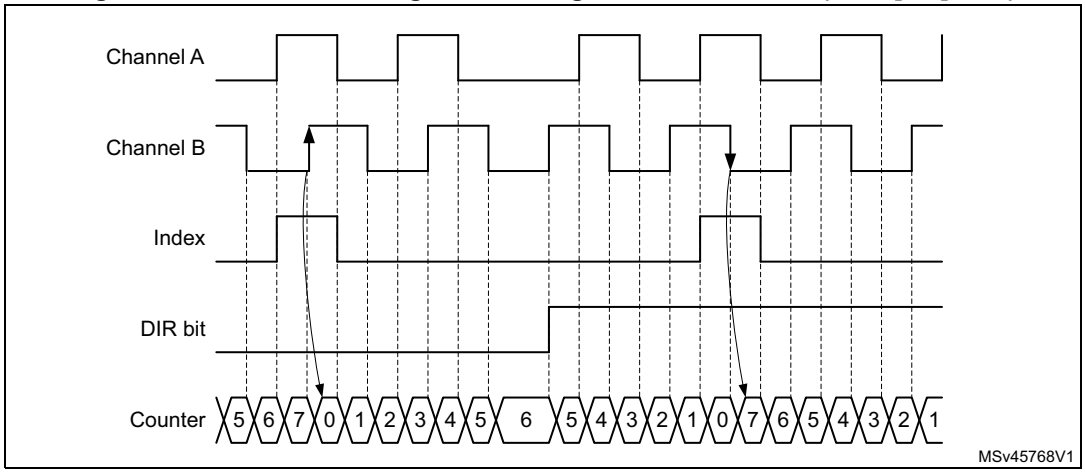


Figure 490 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 490. Counter reading with index ungated (IPOS[1:0] = 00)

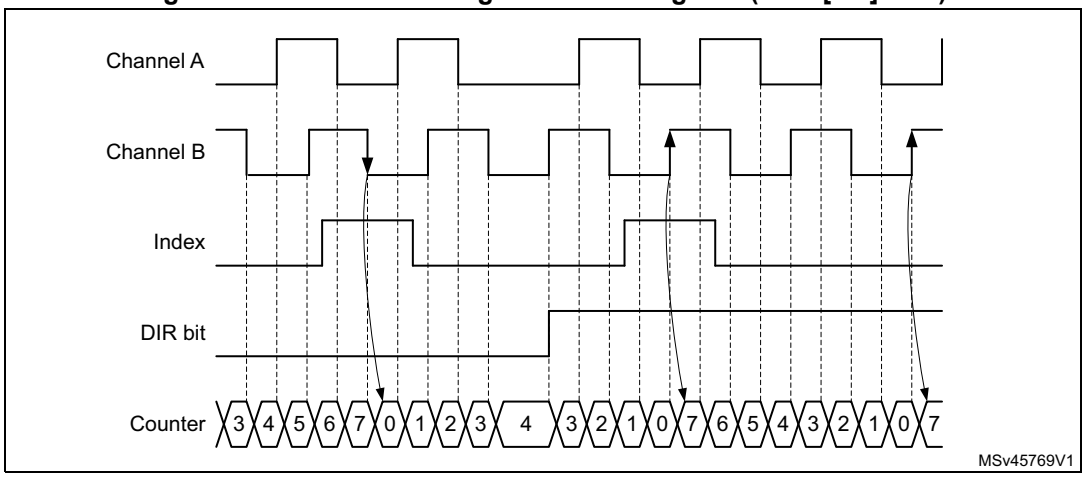


Figure 491 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 491. Counter reading with index gated on channel A and B

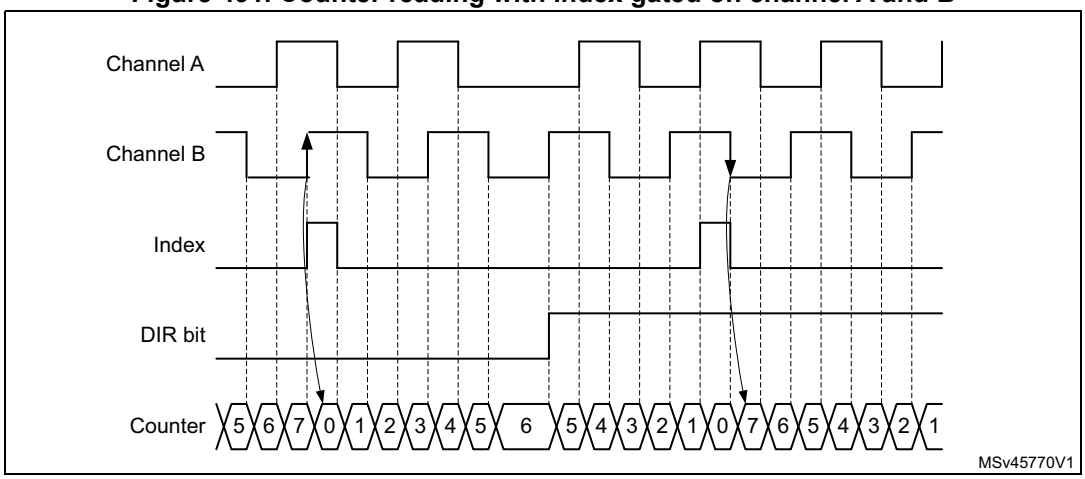


Figure 492 and *Figure 493* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 492. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

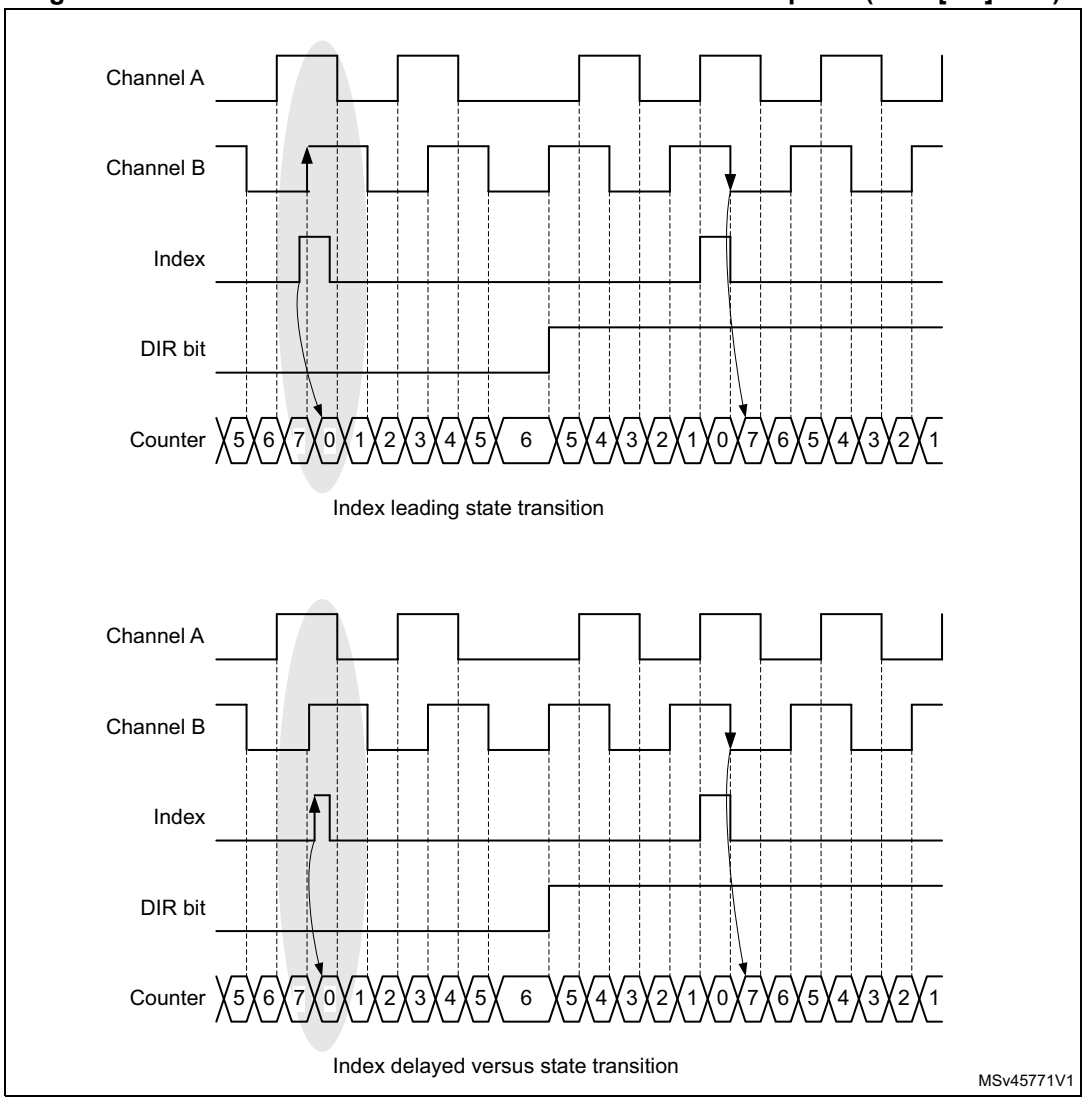


Figure 493. Counter reset Narrow index pulse (closer view, ARR = 0x07)

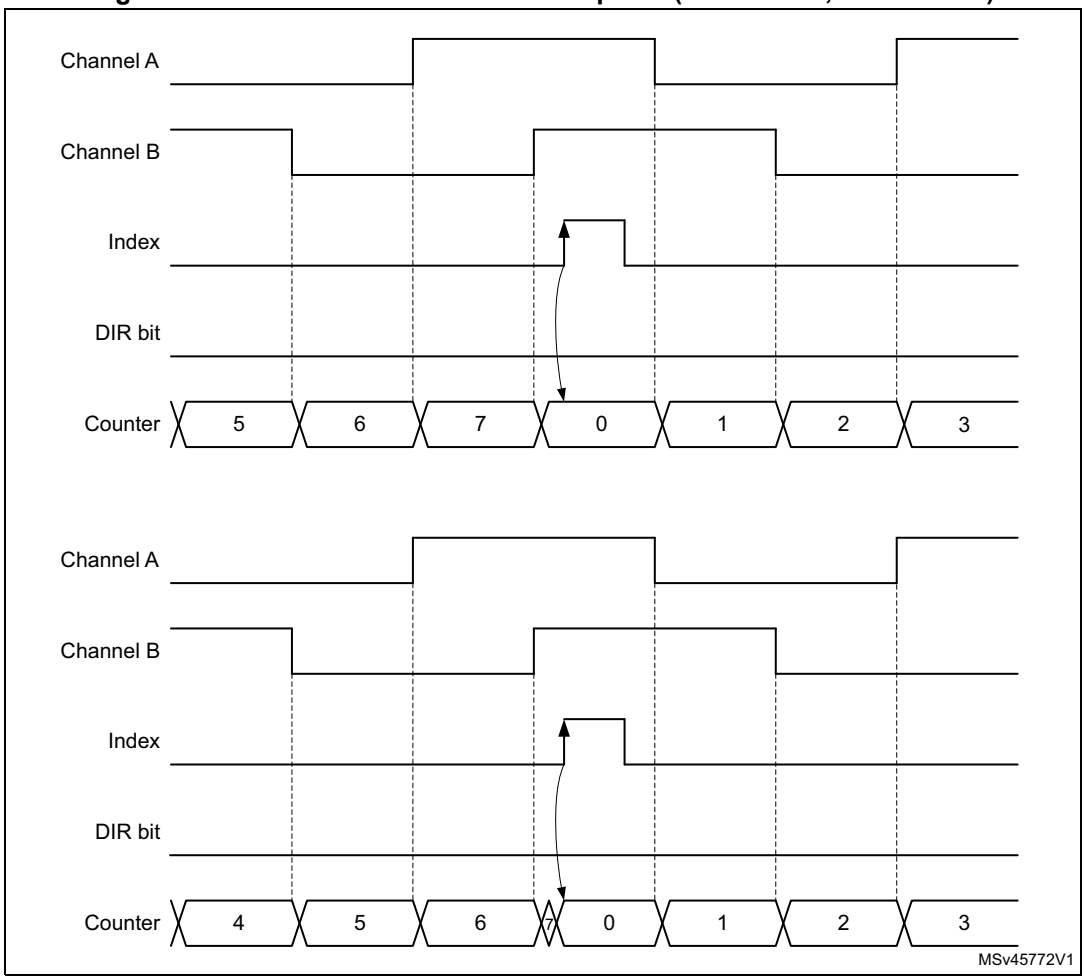
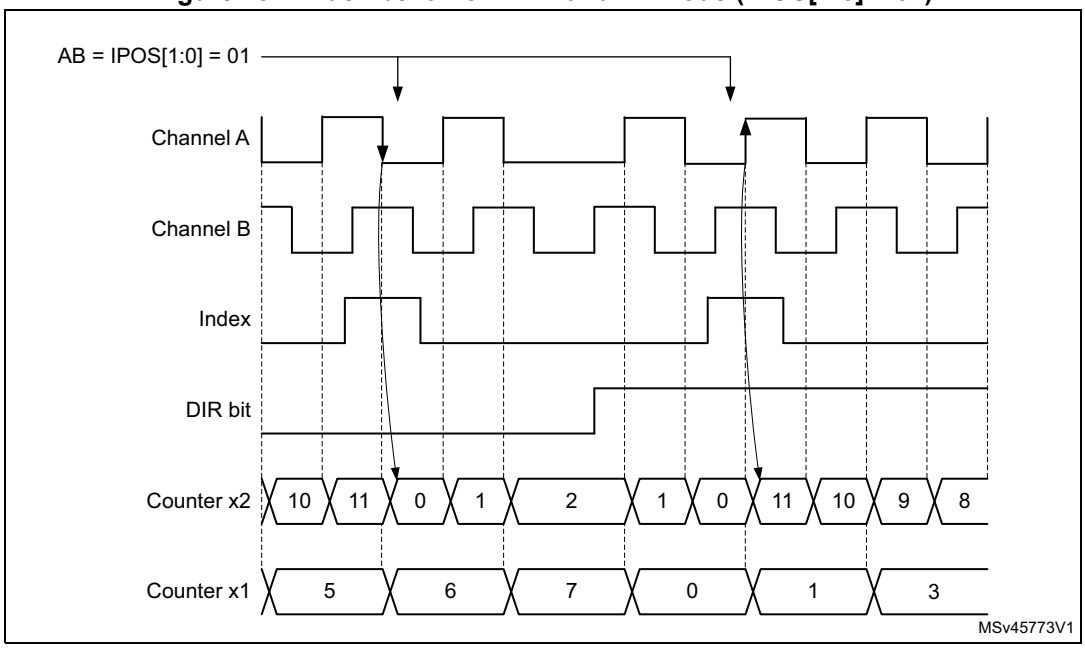


Figure 494 shows how the index is managed in x1 and x2 modes.

Figure 494. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

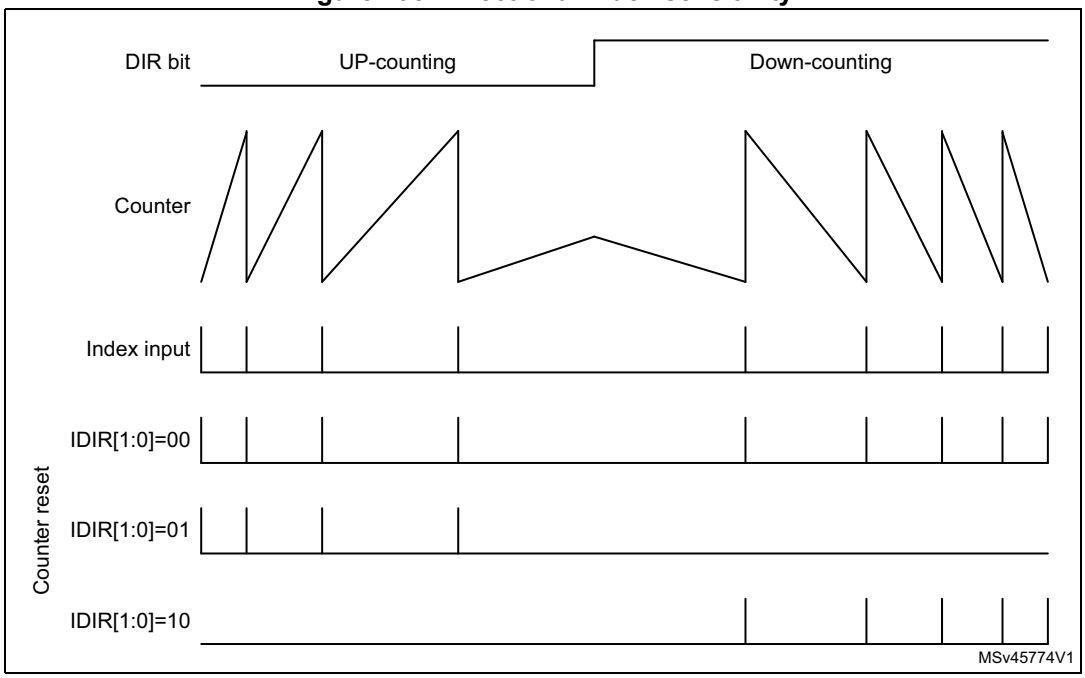


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 495 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

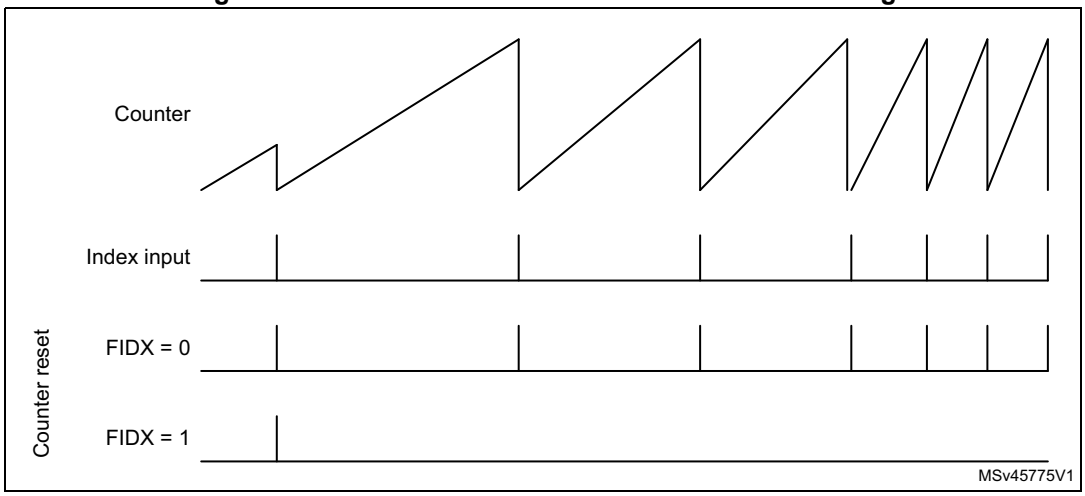
Figure 495. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 496*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 496. Counter reset as function of FIDX bit setting



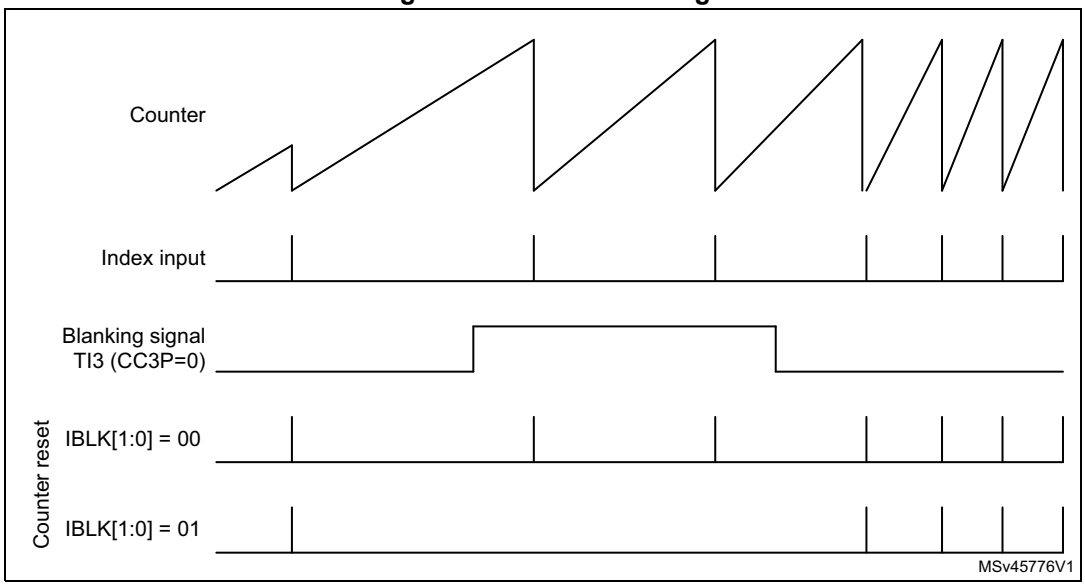
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 497*.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 497. Index blanking



Index management in nonquadrature mode

Figure 498 and *Figure 499* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 498. Index behavior in clock + direction mode, IPOS[0] = 1

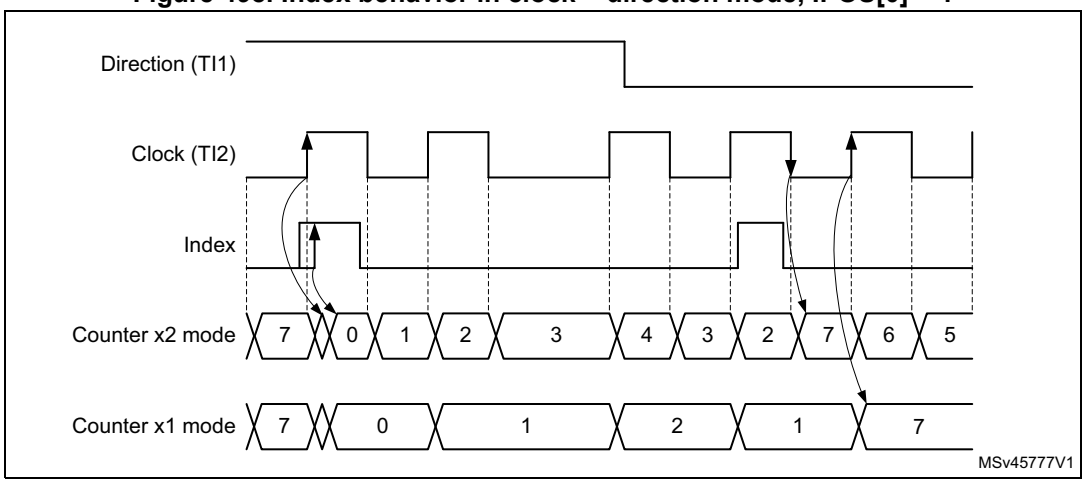
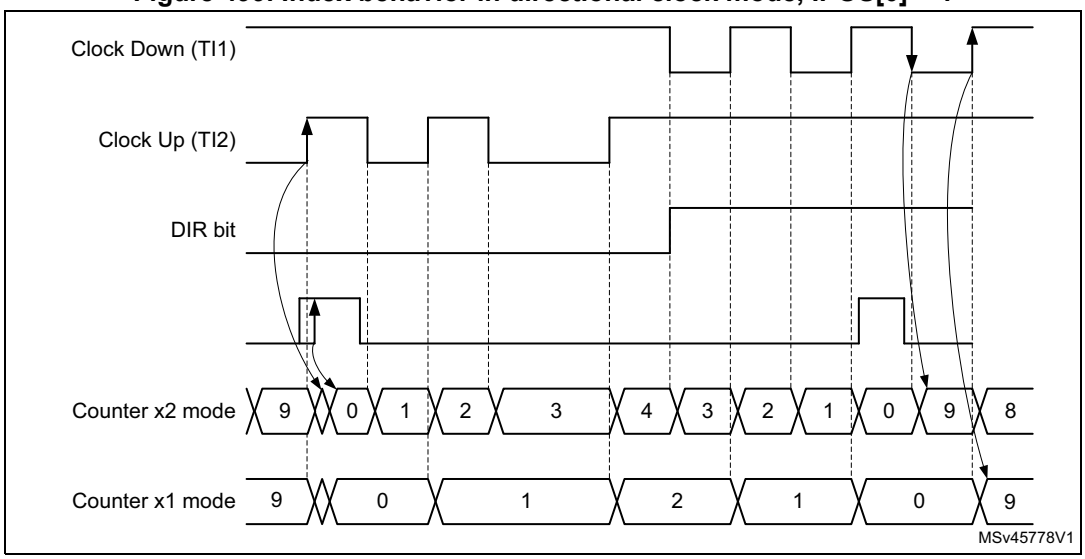


Figure 499. Index behavior in directional clock mode, IPOS[0] = 1

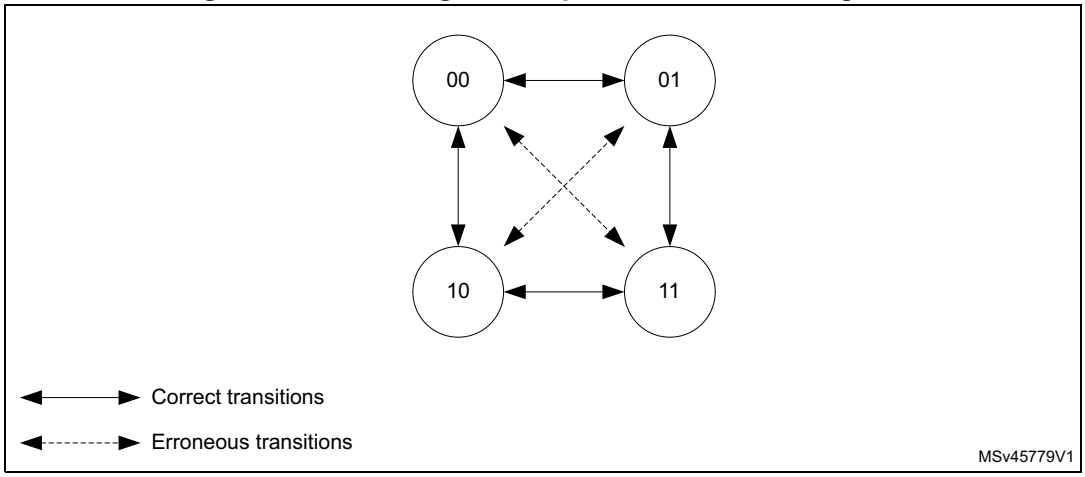


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 500*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 500. State diagram for quadrature encoded signals



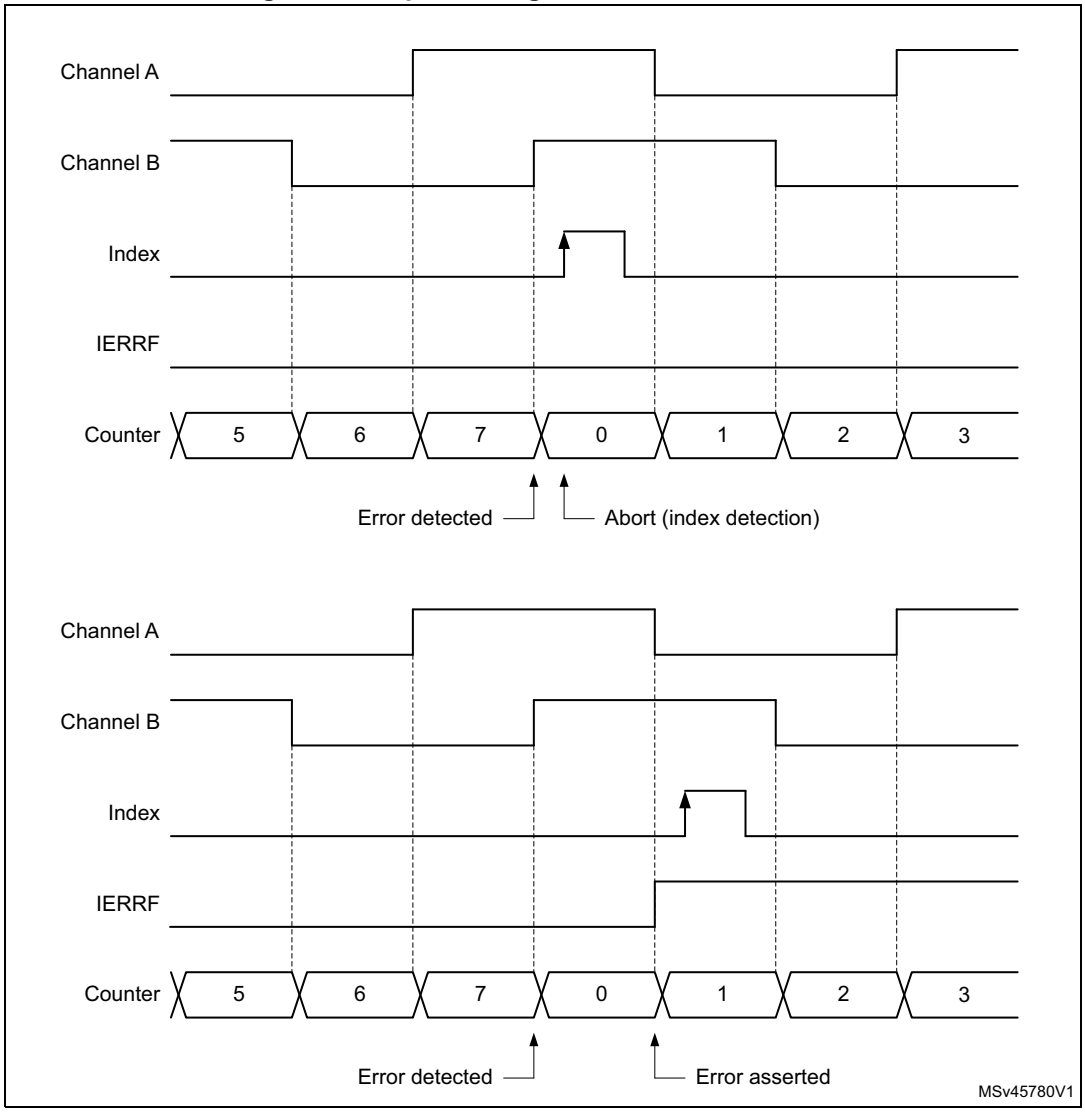
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

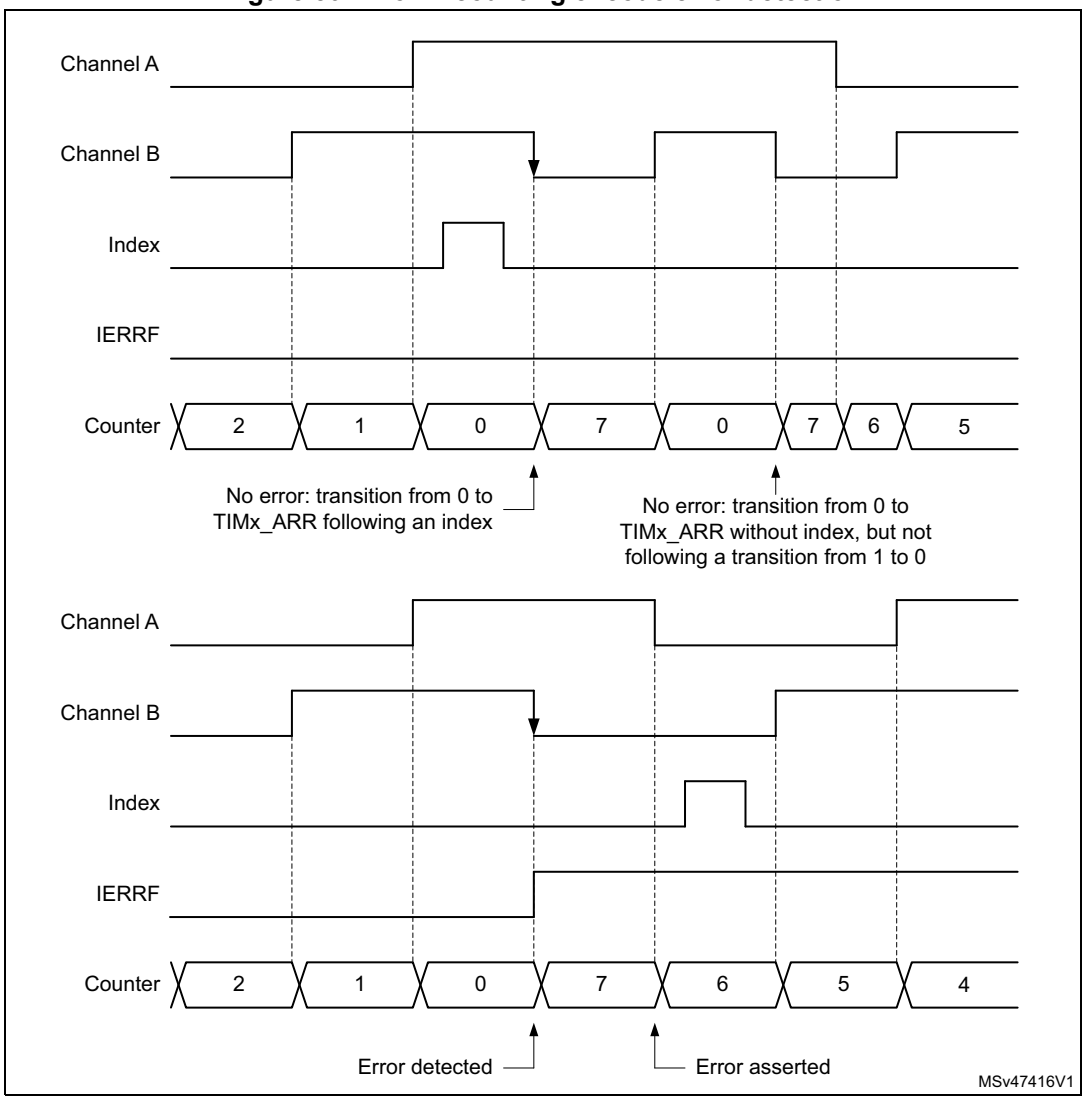
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 501*.

Figure 501. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 502*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 502. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

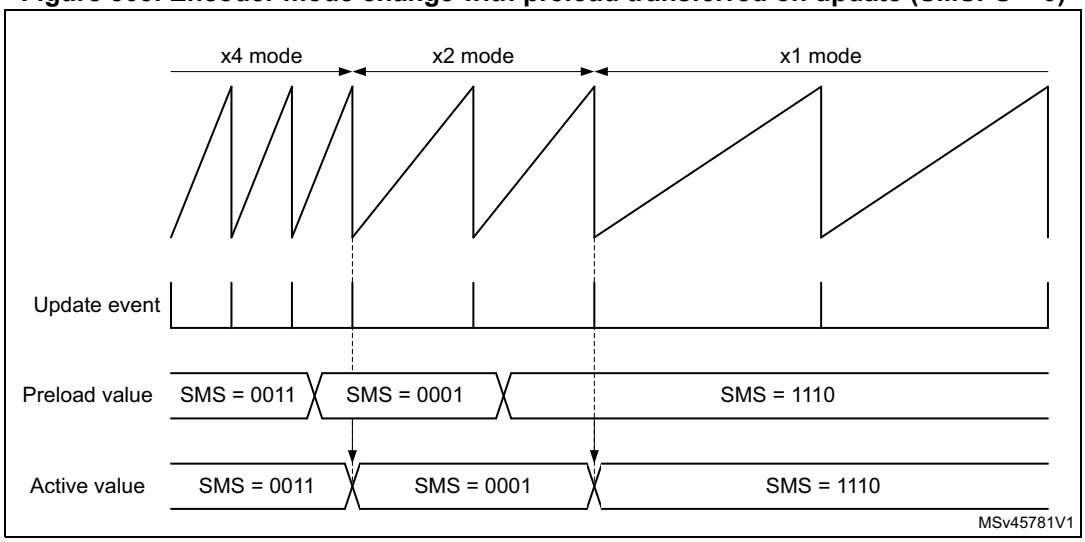
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 503*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 503. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

39.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

39.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

39.4.21 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 38.3.29: Interfacing with Hall sensors*.

39.4.22 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

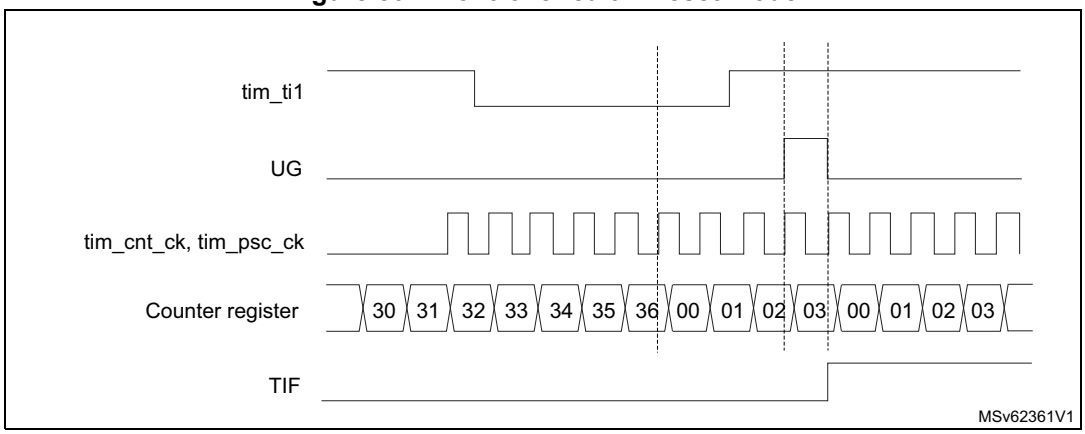
1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the

trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 504. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

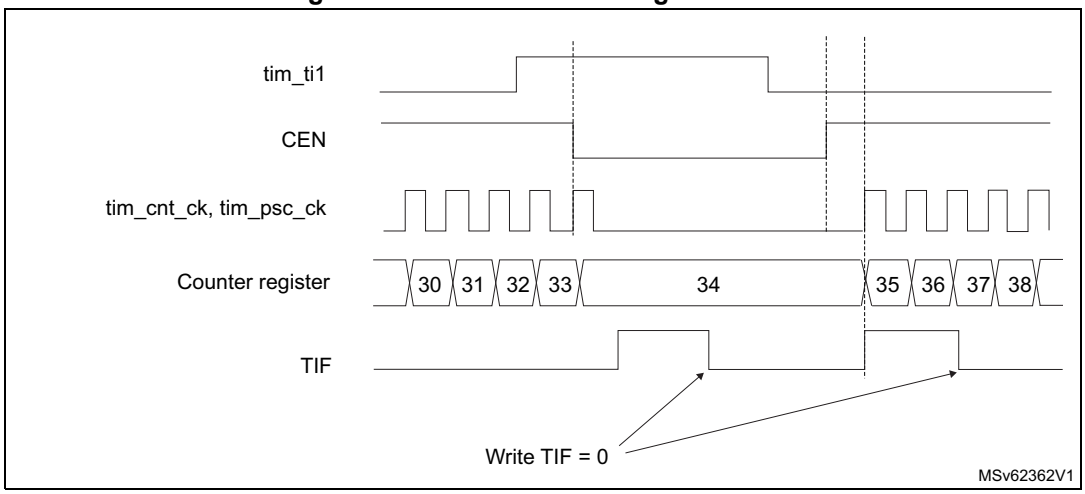
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 505. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

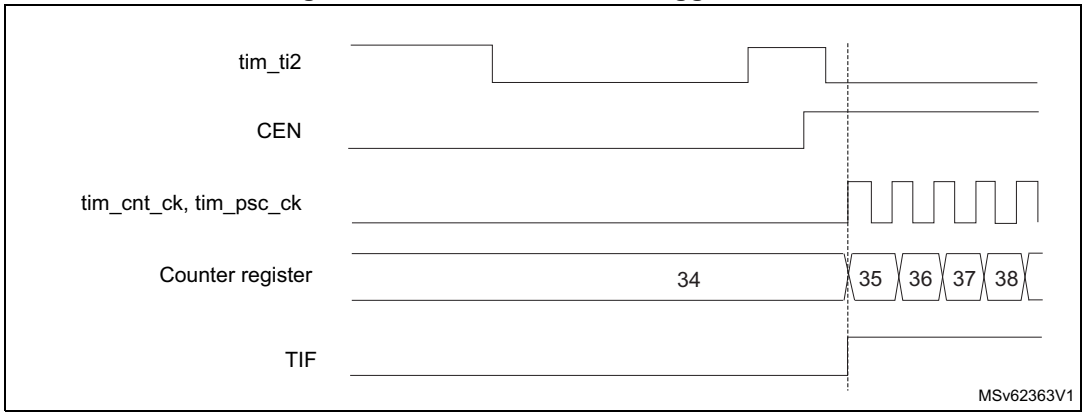
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 506. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

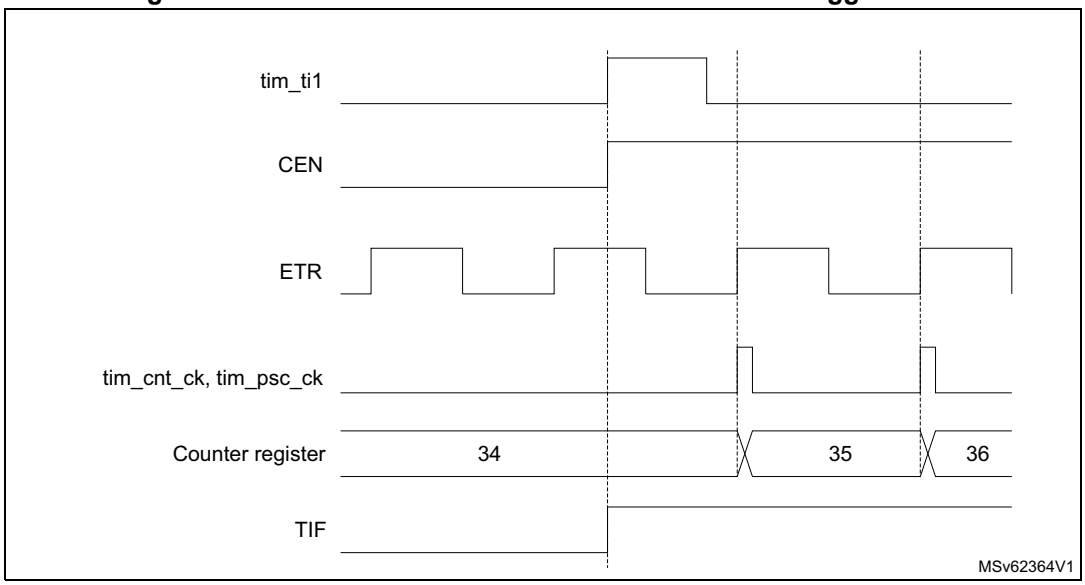
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 507. Control circuit in external clock mode 2 + trigger mode



39.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in Master mode, it can reset, start, stop, or clock the counter of another timer configured in Slave mode.

Figure 508 and *Figure 509* show examples of master/slave timer connections.

Figure 508. Master/Slave timer example

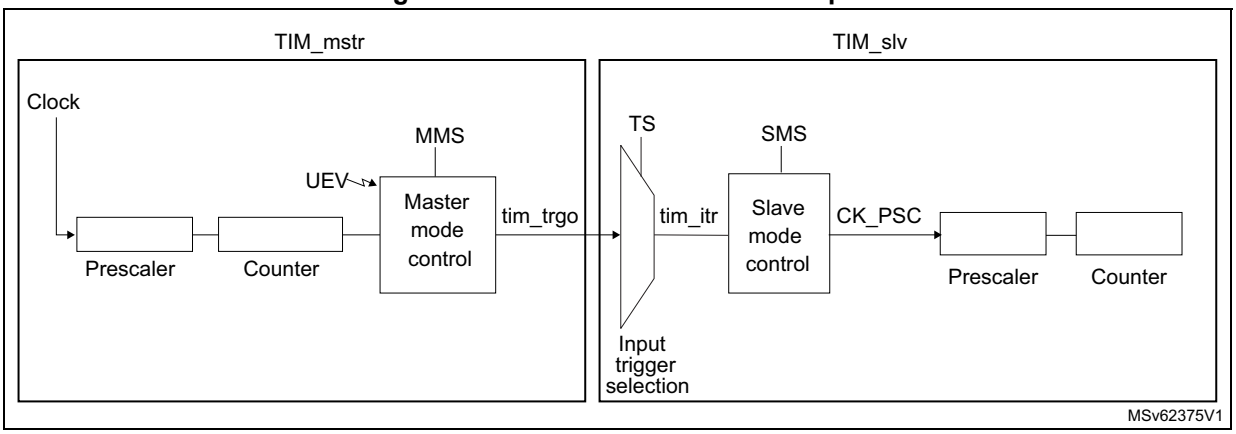
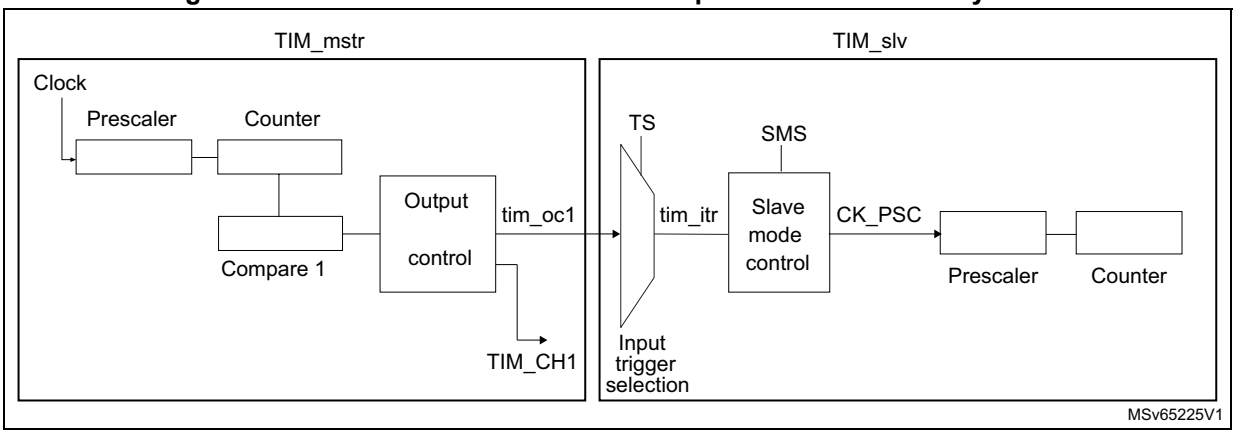


Figure 509. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 509) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals).
The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 508*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

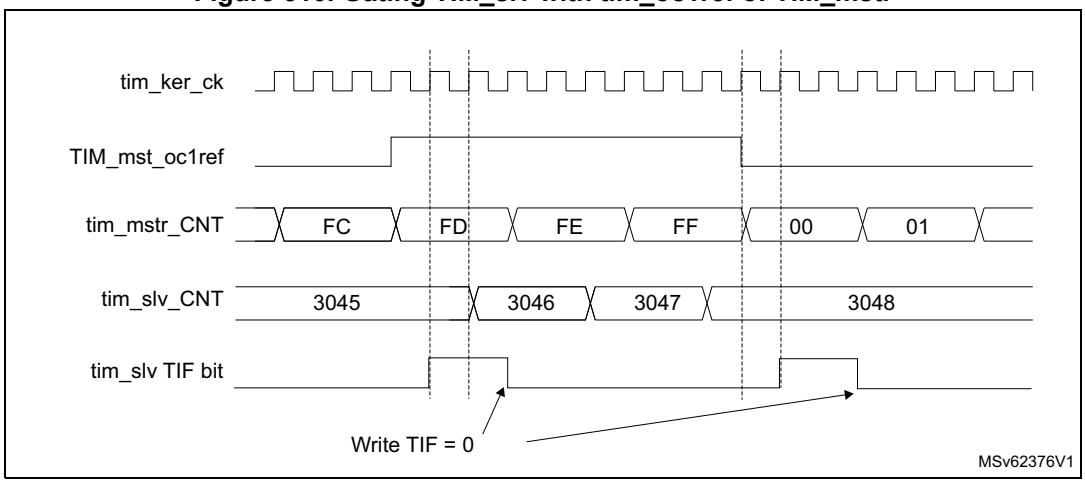
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 508* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 510. Gating TIM_slv with tim_oc1ref of TIM_mstr

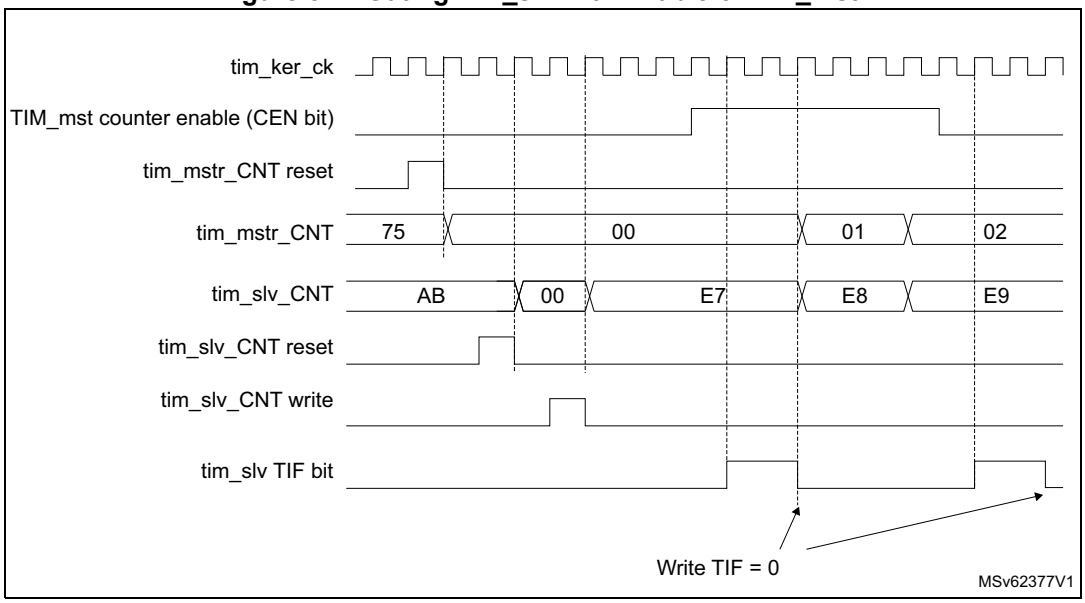


In the example in *Figure 510*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 511*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 511. Gating TIM_slv with Enable of TIM_mstr

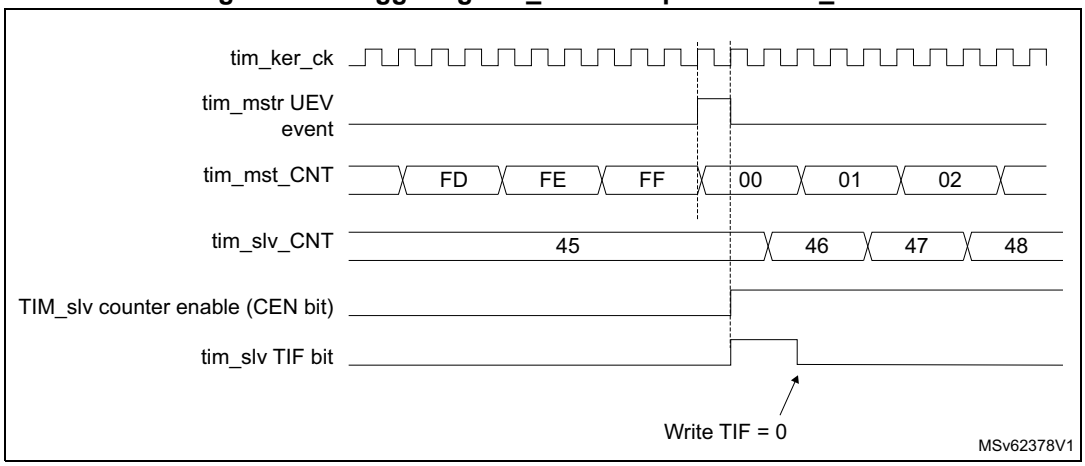


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 508* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

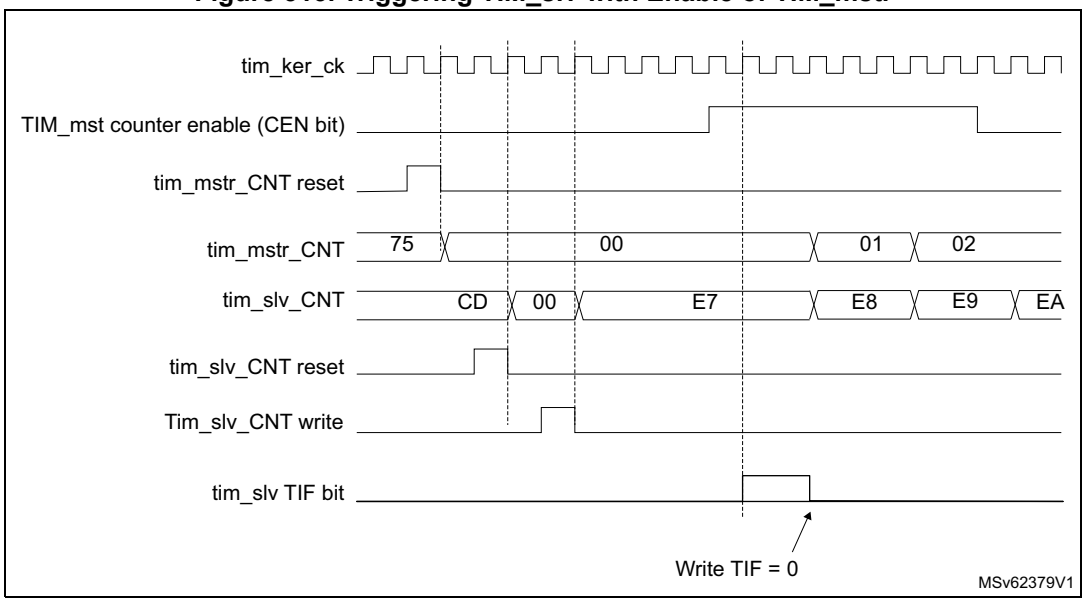
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 512. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 513* shows the behavior with the same configuration as in *Figure 512* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 513. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

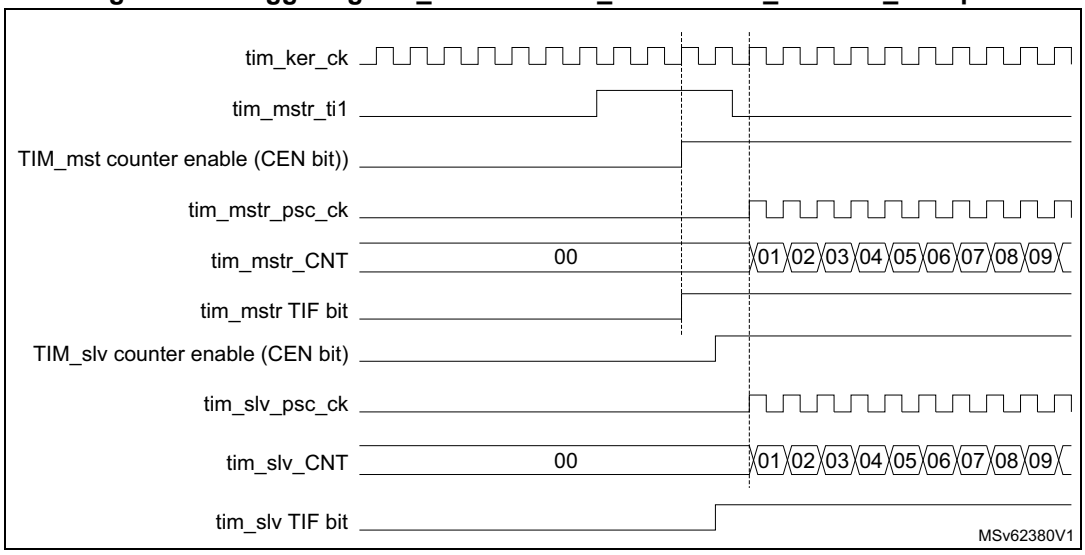
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 508* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 514. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

39.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

39.4.25 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 39.5.29: TIMx DMA control register (TIMx_DCR)(x = 2 to 5)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5,

and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

39.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 427*.

Table 427. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

39.4.27 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

39.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 428. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

39.4.29 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 429*.

Table 429. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

39.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

39.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$

01: $t_{DTS} = 2 \times t_{tim_ker_ck}$

10: $t_{DTS} = 4 \times t_{tim_ker_ck}$

11: Reserved

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

CEN is cleared automatically in one-pulse mode, when an update event occurs.

39.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

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

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input
1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 38.3.29: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

39.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etrp frequency divided by 2

10: tim_etrp frequency divided by 4

11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 39.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000:Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001:Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010:Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011:Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100:Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101:Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110:Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111:External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000:Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001:Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010:Encoder mode: Clock plus direction, x2 mode.

1011:Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100:Encoder mode: Directional clock, x2 mode.

1101:Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110:Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111:Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

39.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

39.5.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

39.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

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

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (down-counting).

39.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000:No filter, sampling is done at f_{DTS}

0001:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011:$f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100:$f_{SAMPLING} = f_{DTS}/2$, N = 6

0101:$f_{SAMPLING} = f_{DTS}/2$, N = 8

0110:$f_{SAMPLING} = f_{DTS}/4$, N = 6

0111:$f_{SAMPLING} = f_{DTS}/4$, N = 8

1000:$f_{SAMPLING} = f_{DTS}/8$, N = 6

1001:$f_{SAMPLING} = f_{DTS}/8$, N = 8

1010:$f_{SAMPLING} = f_{DTS}/16$, N = 5

1011:$f_{SAMPLING} = f_{DTS}/16$, N = 6

1100:$f_{SAMPLING} = f_{DTS}/16$, N = 8

1101:$f_{SAMPLING} = f_{DTS}/32$, N = 5

1110:$f_{SAMPLING} = f_{DTS}/32$, N = 6

1111:$f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

39.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In up-counting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In down-counting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

39.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

39.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0]

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000:Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001:Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010:Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011:Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100:Force inactive level - tim_oc3ref is forced low.

0101:Force active level - tim_oc3ref is forced high.

0110:PWM mode 1 - In up-counting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111:PWM mode 2 - In up-counting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In down-counting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000:Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001:Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010:Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011:Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100:Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110:Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111:Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

39.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1:non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:this configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 430. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

39.5.12 TIMx counter (TIMx_CNT)(x = 3, 4)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
Reserved
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[15:0]. The fractional part is not available.

39.5.13 TIMx counter (TIMx_CNT)(x = 2, 5)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

39.5.14 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

39.5.15 TIMx autoreload register (TIMx_ARR)(x = 3, 4)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Low autoreload value

ARR is the value to be loaded in the actual autoreload register.
Refer to the *Section 39.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the autoreload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the autoreload value.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

39.5.16 TIMx autoreload register (TIMx_ARR)(x = 2, 5)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

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

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 39.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

39.5.17 TIMx capture/compare register 1 (TIMx_CCR1)(x = 3, 4)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR1[15:0] bits hold the capture value. The CCR1[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:0]. The CCR1[3:0] bits are reset.

39.5.18 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2, 5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

39.5.19 TIMx capture/compare register 2 (TIMx_CCR2)(x = 3, 4)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 1 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR2[15:0] bits hold the capture value. The CCR2[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:0]. The CCR2[3:0] bits are reset.

39.5.20 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2, 5)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

39.5.21 TIMx capture/compare register 3 (TIMx_CCR3)(x = 3, 4)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR3[15:0] bits hold the capture value. The CCR3[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[19:0]. The CCR3[3:0] bits are reset.

39.5.22 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2, 5)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

39.5.23 TIMx capture/compare register 4 (TIMx_CCR4)(x = 3, 4)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The CCR4[15:0] bits hold the capture value. The CCR4[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:0]. The CCR4[3:0] bits are reset.

39.5.24 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2, 5)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

39.5.25 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)

Address offset: 0x058

Reset value: 0x0000 0000

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

39.5.26 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

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

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

39.5.27 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)

Address offset: 0x060

Reset value: 0x0000 0000

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

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

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

39.5.28 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 39.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

39.5.29 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x3DC

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

39.5.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

39.5.31 TIMx register map

TIMx registers are mapped as described in the table below.

Table 431. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value							0																		0	0	0	0	0			
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value								0	0	0	0							0		0	0	0	0	0		0		0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value								0	0	0	0									0	0	0	0			0		0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 431. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	TIMx_CNT	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR (x = 3, 4)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x02C	TIMx_ARR (x = 2, 5)	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved	Reserved																															
0x034	TIMx_CCR1	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x064	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x068.. 0x3D8	Reserved	Res.																															

Table 431. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

40 Basic timers (TIM6/TIM7)

40.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The basic timer can also be used for triggering the digital-to-analog converter. This is done with the trigger output of the timer.

The timers are completely independent, and do not share any resources.

40.2 TIM6/TIM7 main features

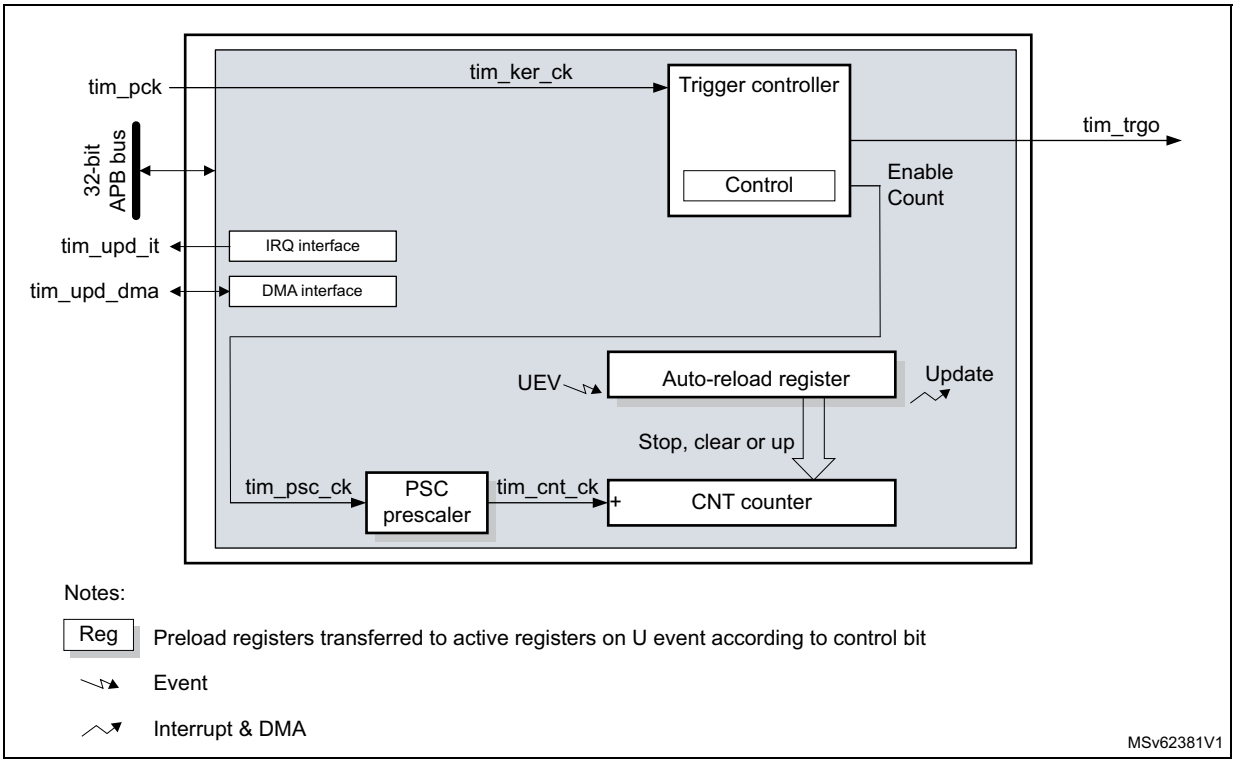
Basic timer (TIM6/TIM7) features include:

- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Synchronization circuit to trigger the DAC.
- Interrupt/DMA generation on the update event: counter overflow.

40.3 TIM6/TIM7 functional description

40.3.1 TIM6/TIM7 block diagram

Figure 515. Basic timer block diagram



40.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 432. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals (DAC).
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

40.3.3 TIM6/TIM7 clocks

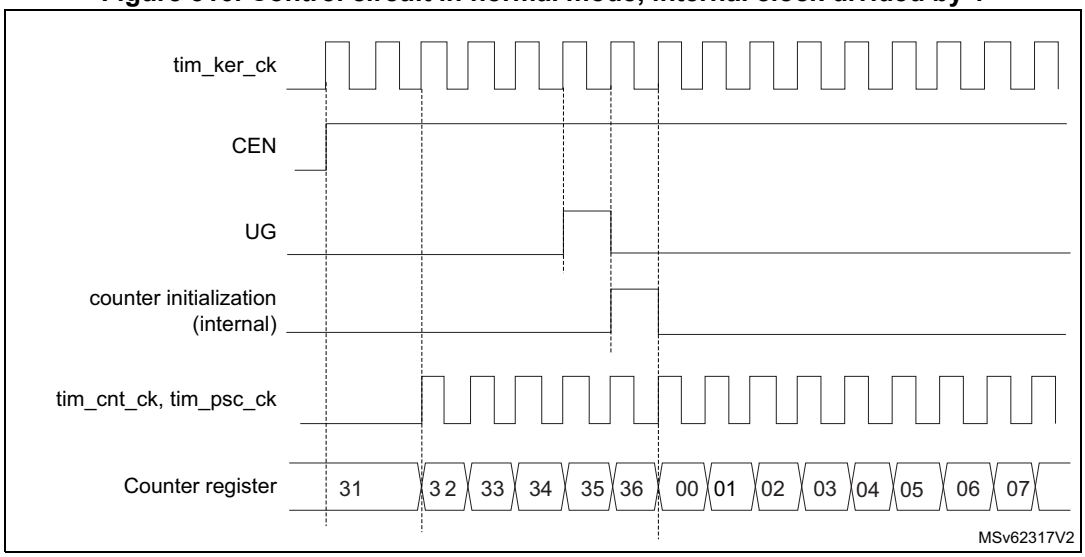
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 516 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 516. Control circuit in normal mode, internal clock divided by 1



40.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-Reload register (TIMx_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 517 and *Figure 518* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 517. Counter timing diagram with prescaler division change from 1 to 2

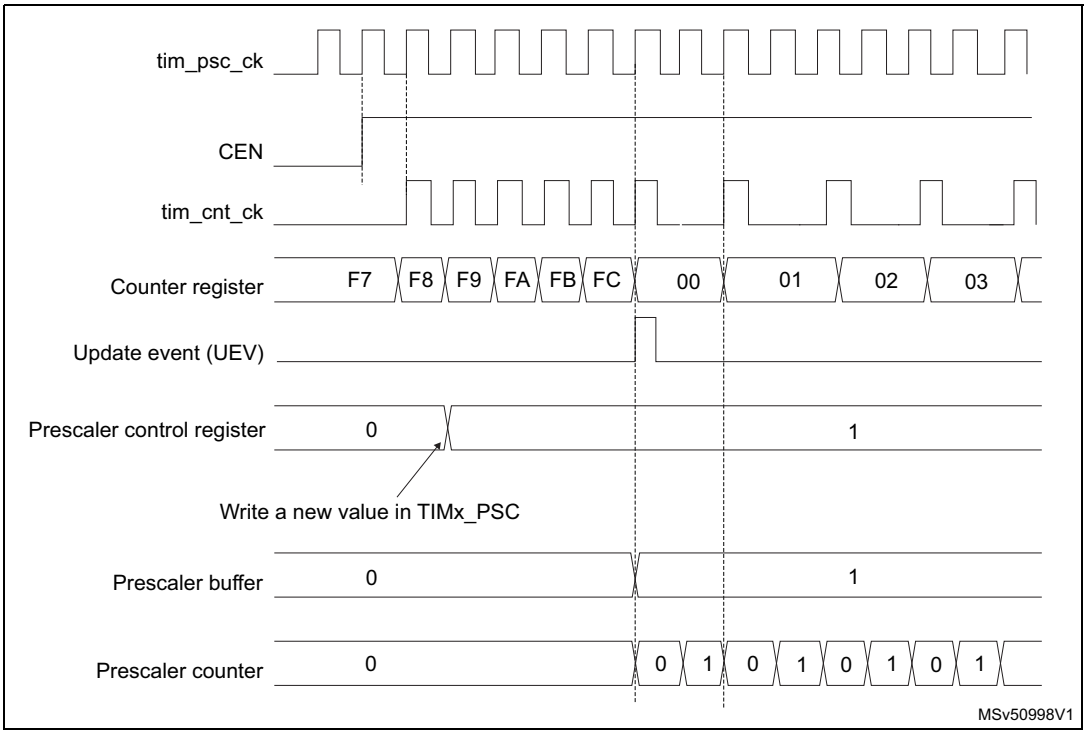
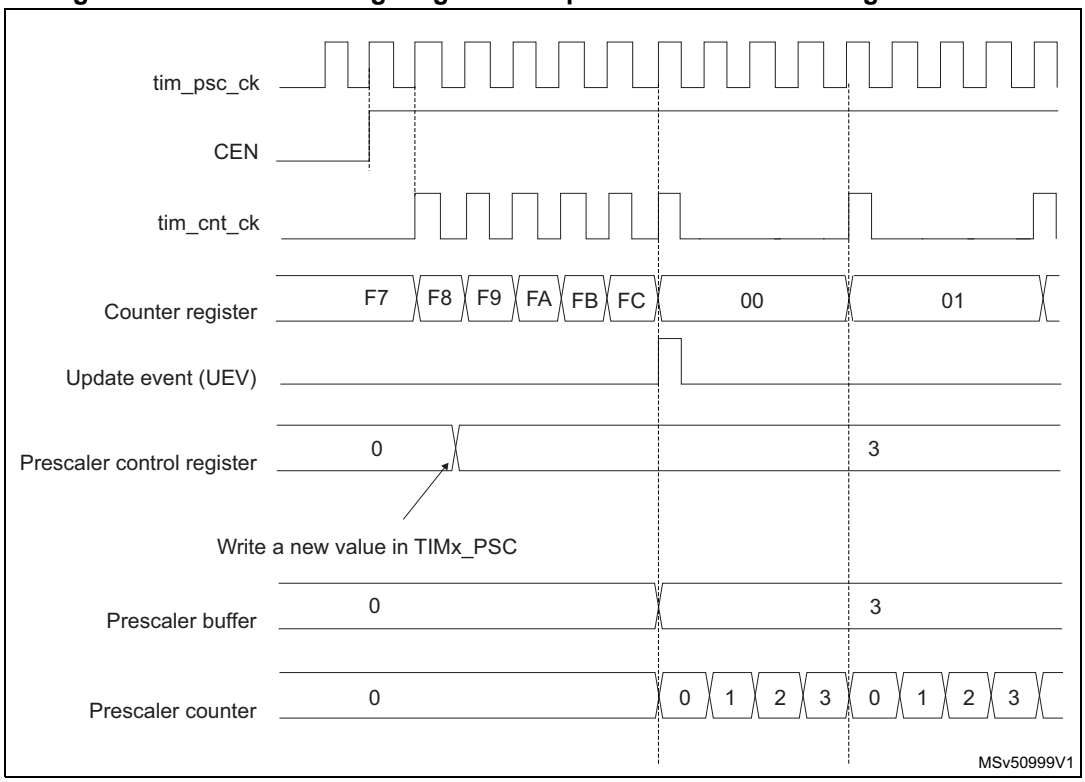


Figure 518. Counter timing diagram with prescaler division change from 1 to 4



40.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 519. Counter timing diagram, internal clock divided by 1

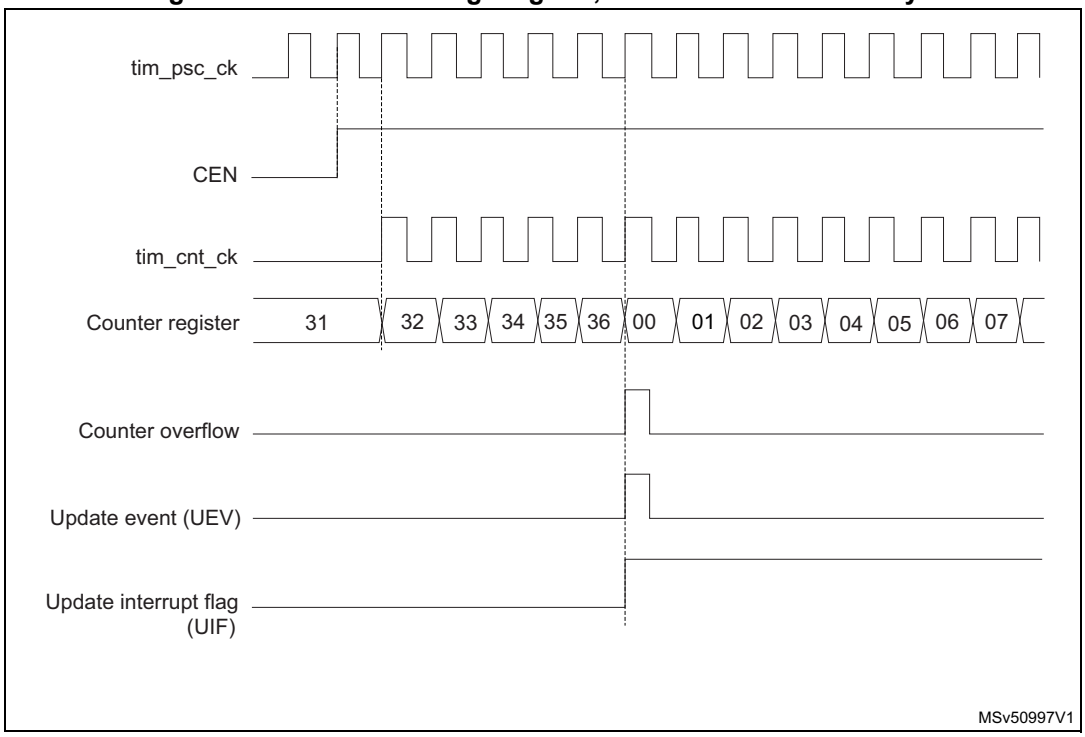


Figure 520. Counter timing diagram, internal clock divided by 2

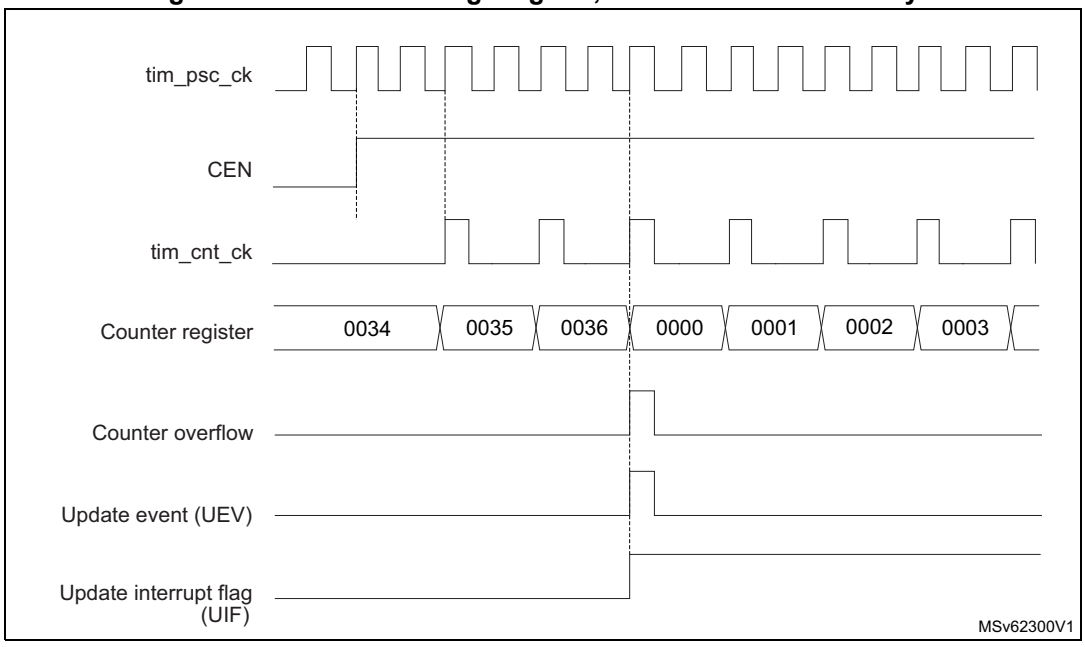


Figure 521. Counter timing diagram, internal clock divided by 4

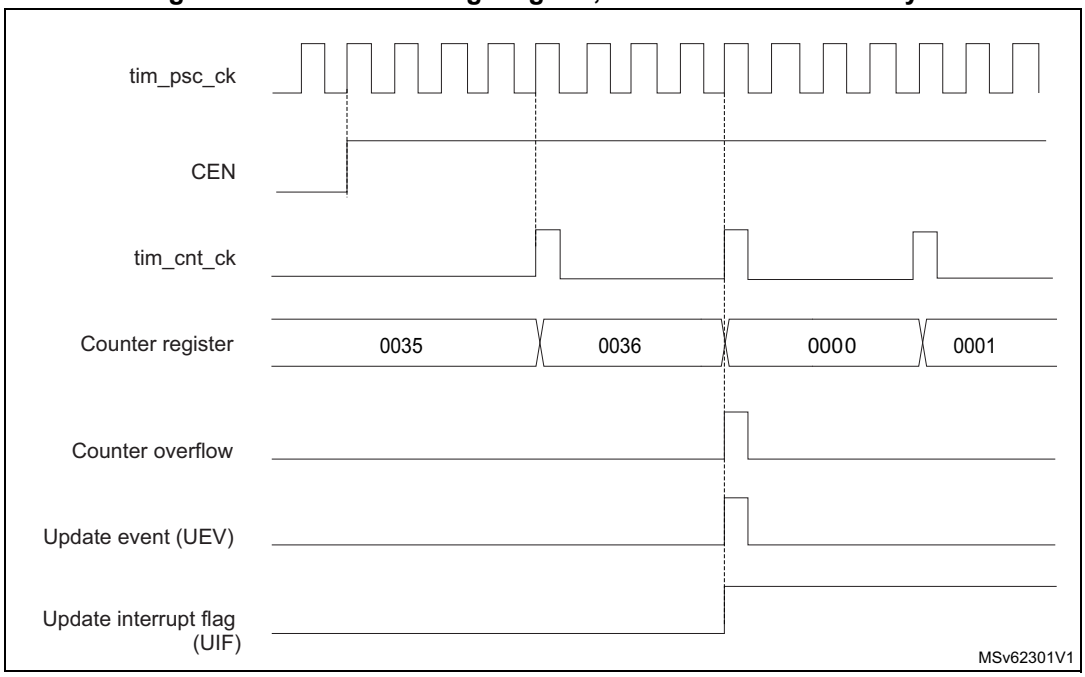


Figure 522. Counter timing diagram, internal clock divided by N

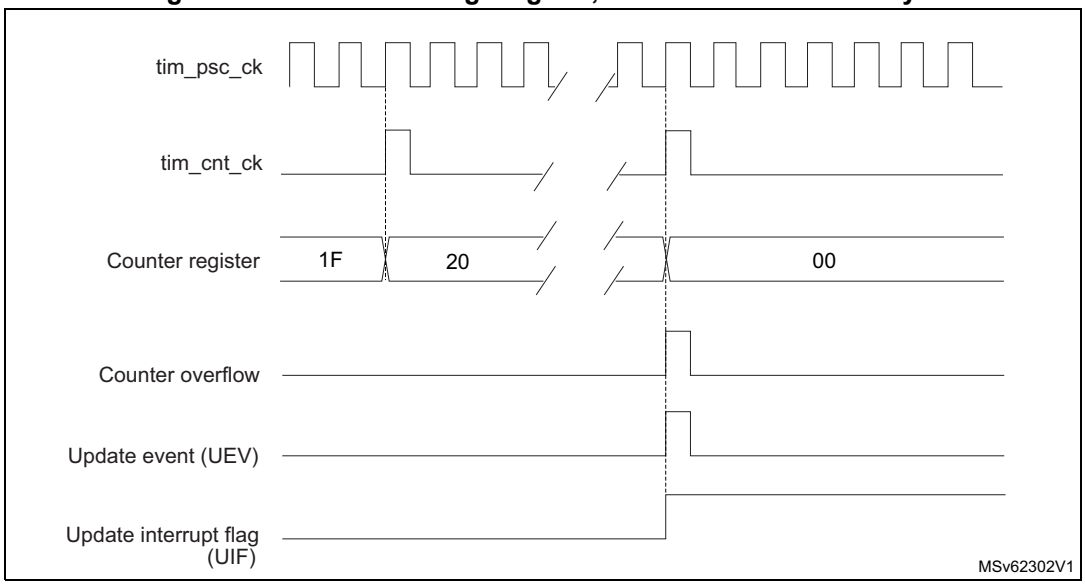


Figure 523. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

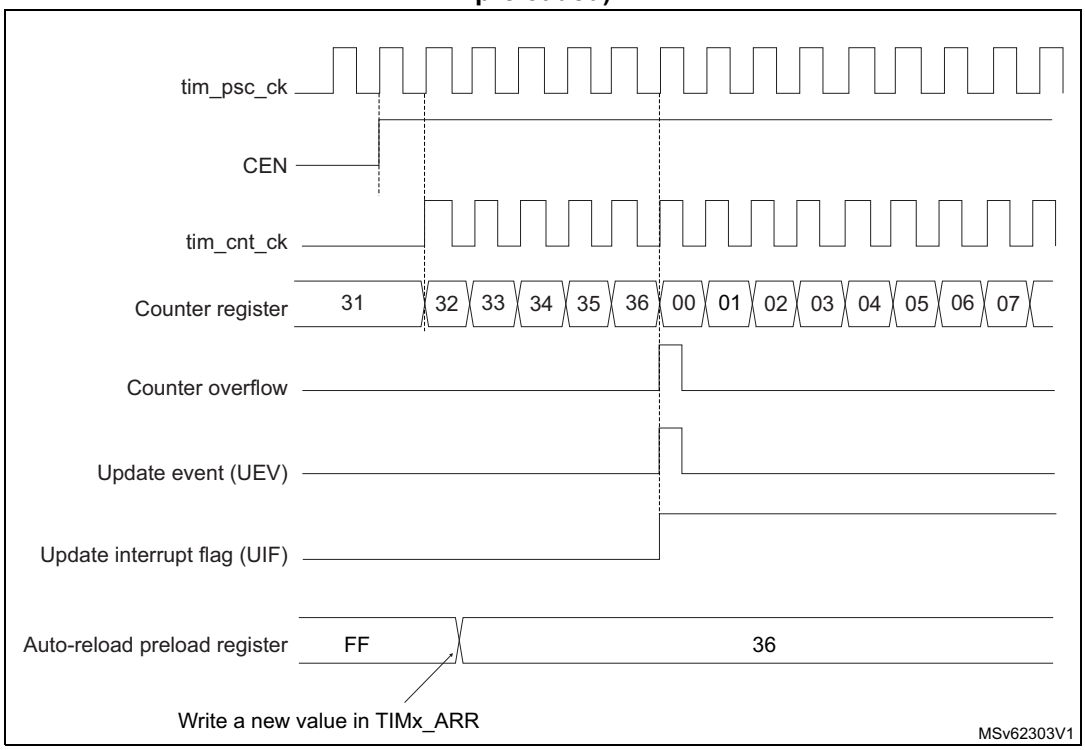
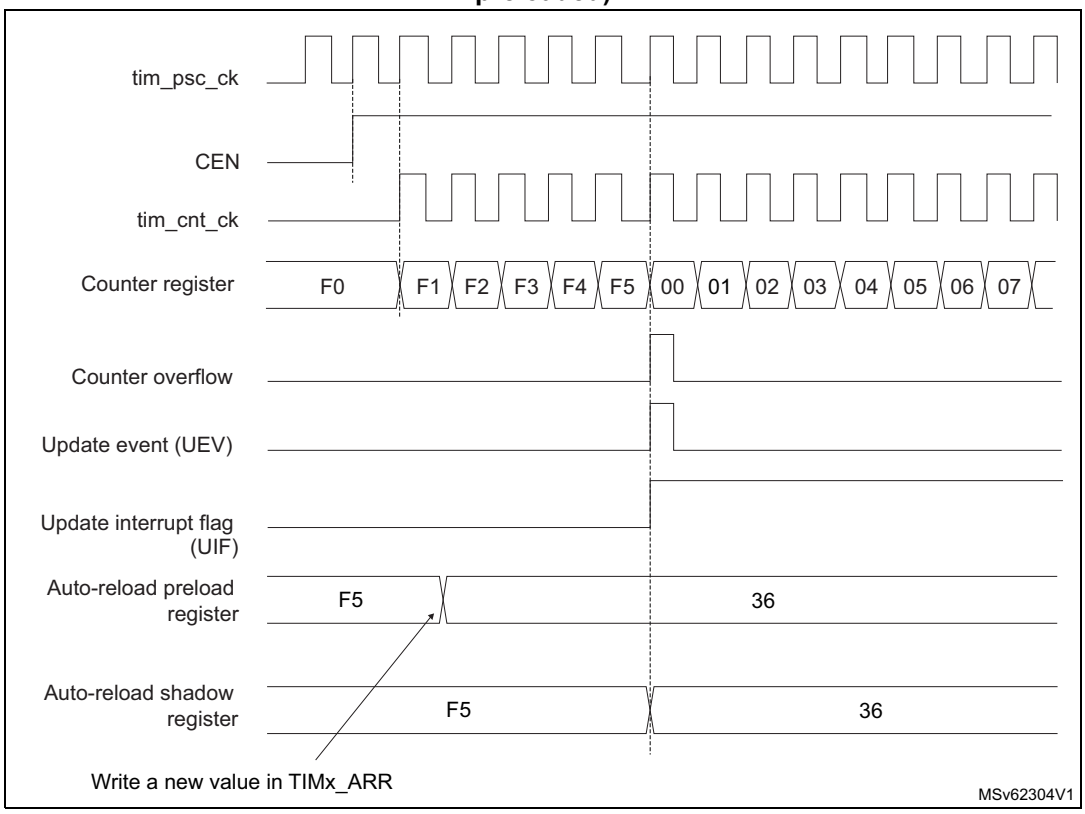


Figure 524. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



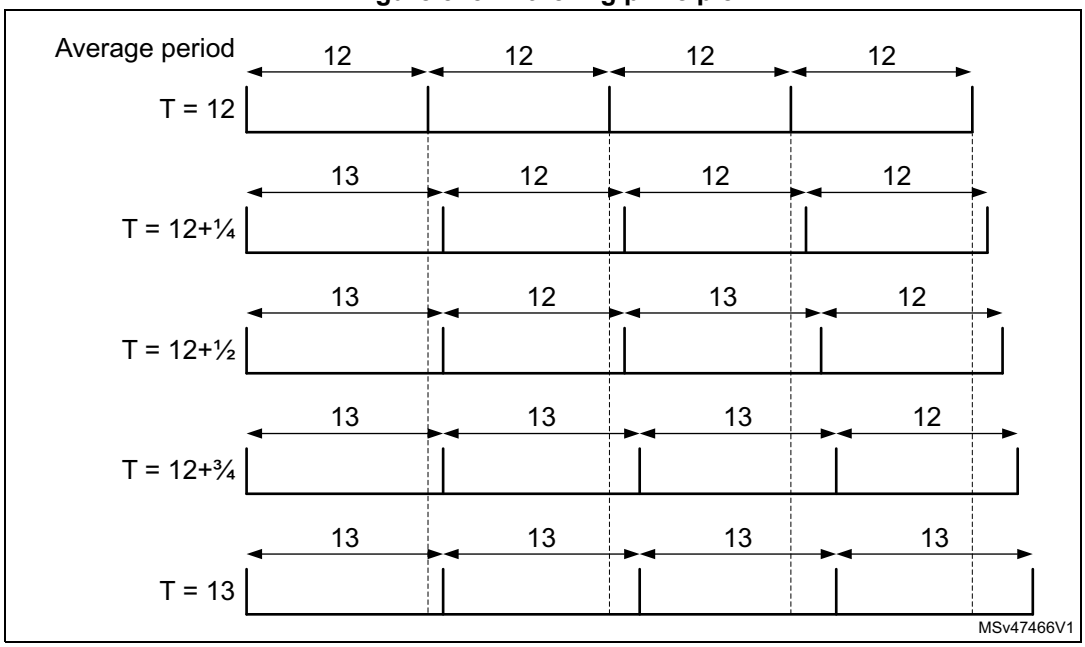
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger (typically for a DAC).

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 525 presents the dithering principle applied to four consecutive counting periods.

Figure 525. Dithering principle

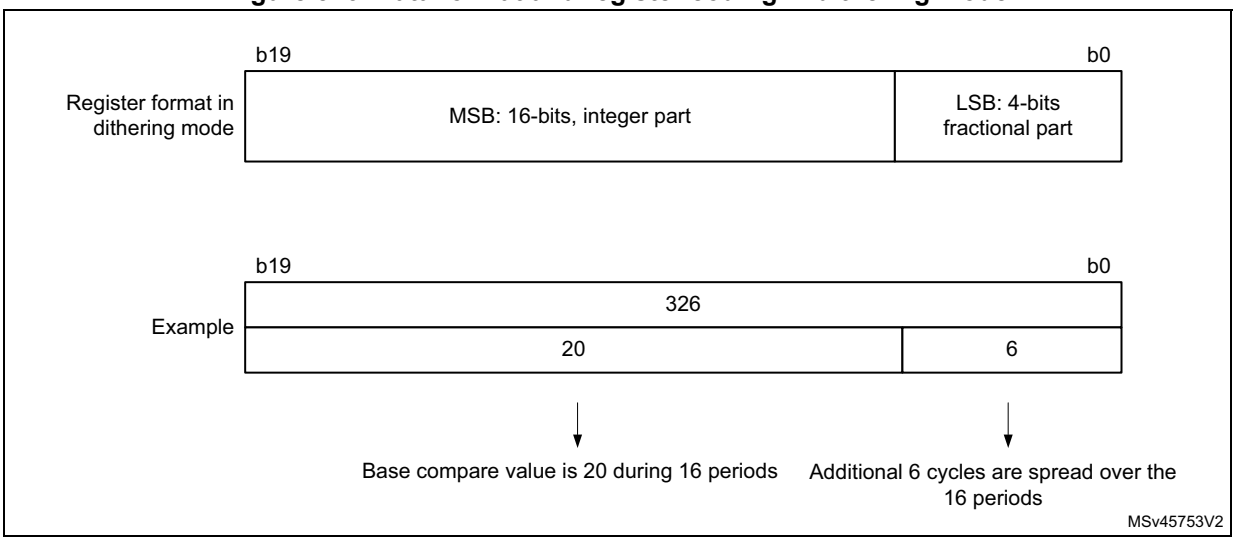


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 526* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 526. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

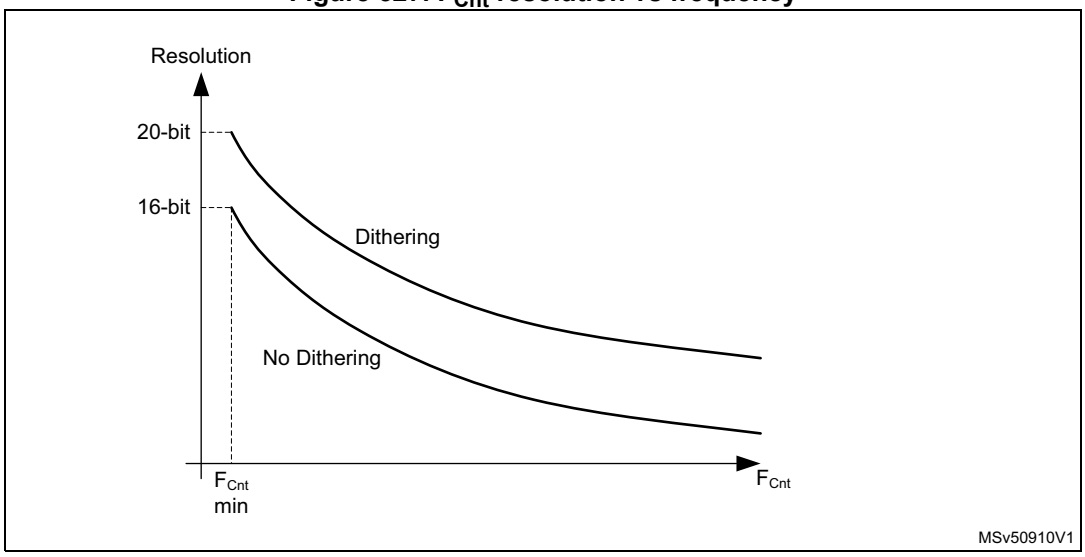
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

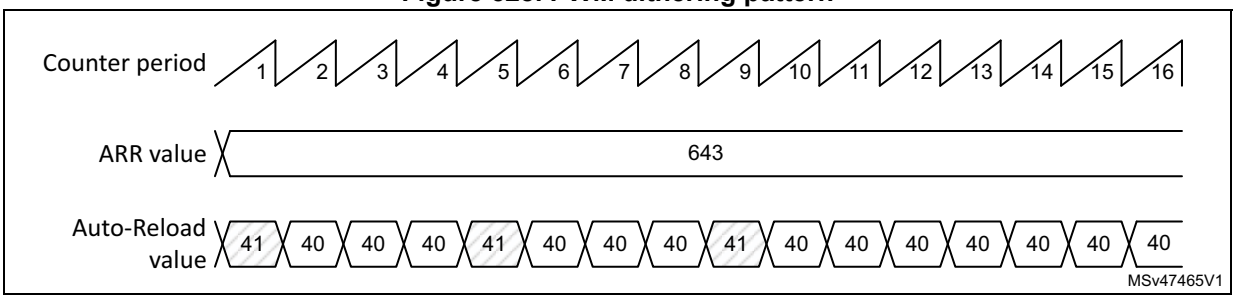
As shown on *Figure 527*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 527. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 528*.

Figure 528. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 433*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 433. TIMx_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

40.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

40.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

40.3.8 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in *Table 434*.

Table 434. DMA request

DMA acronym	DMA request	Enable control bit
tim_upd_dma	Update	UDE

40.3.9 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

40.3.10 TIM6/TIM7 low-power modes

Table 435. Effect of low-power modes on TIM6/TIM7

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

40.3.11 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in *Table 436*.

Table 436. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

40.4 TIM6/TIM7 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

40.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

40.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7)

Address offset: 0x04

Reset value: 0x0000

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

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

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

40.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7)

Address offset: 0x0C

Reset value: 0x0000

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

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled.
1: Update interrupt enabled.

40.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

–On counter overflow if UDIS = 0 in the TIMx_CR1 register.

–When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

40.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14

Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

40.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

40.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.

40.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 40.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

40.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 437. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	**TIMx_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value																										0	0	0				
0x08	Reserved																																
0x0C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	**TIMx_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

41 General-purpose timers (TIM12/TIM13/TIM14)

41.1 TIM12/TIM13/TIM14 introduction

The TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 41.4.20: Timer synchronization (TIM12 only)*.

41.2 TIM12 main features

The features of the TIM12 general-purpose timer include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization, or count by internal trigger)
  - Input capture
  - Output compare

41.3 TIM13/TIM14 main features

The features of general-purpose timers TIM13/TIM14 include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

41.4 TIM12/TIM13/TIM14 functional description

41.4.1 Block diagram

Figure 529. General-purpose timer block diagram (TIM12)

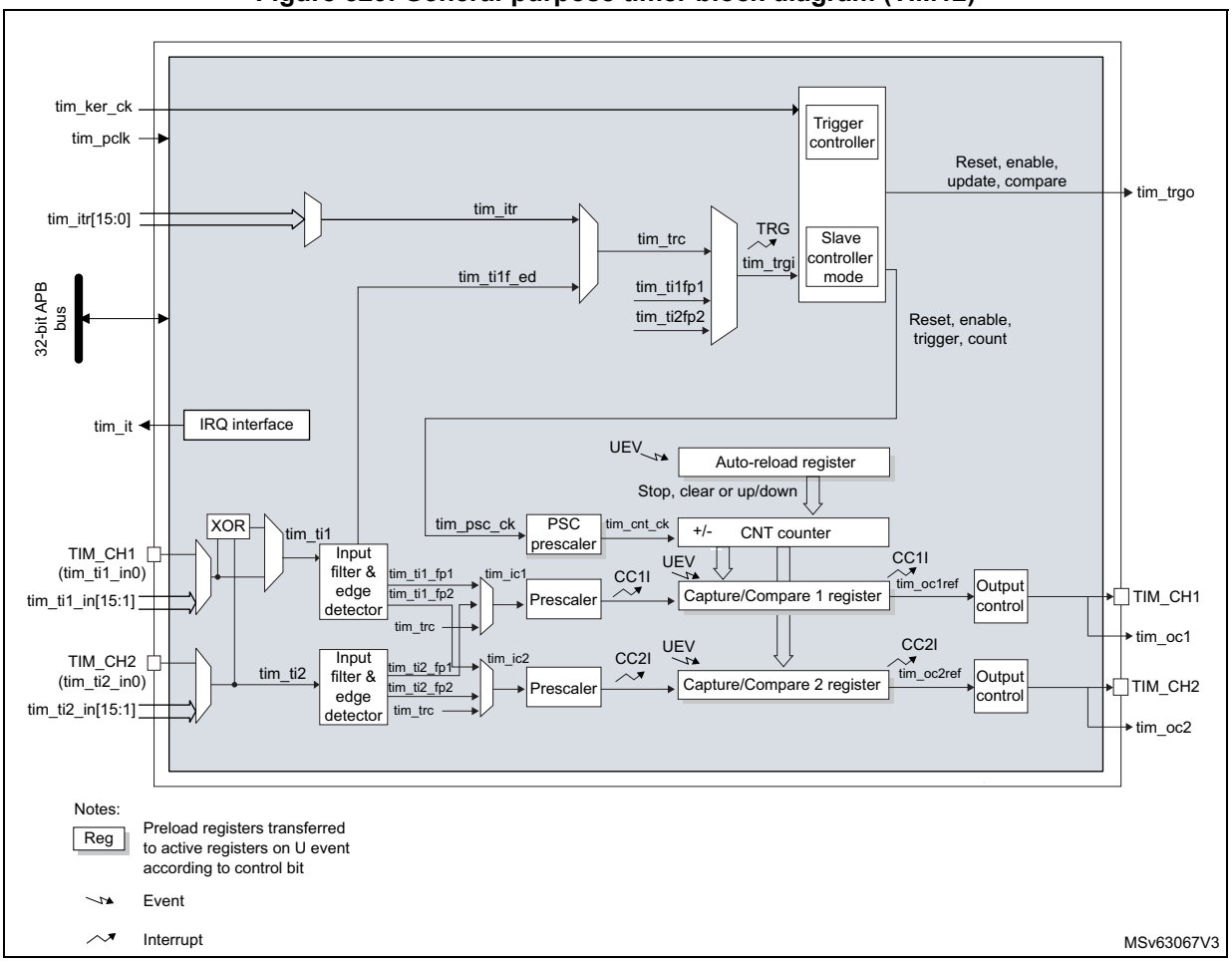
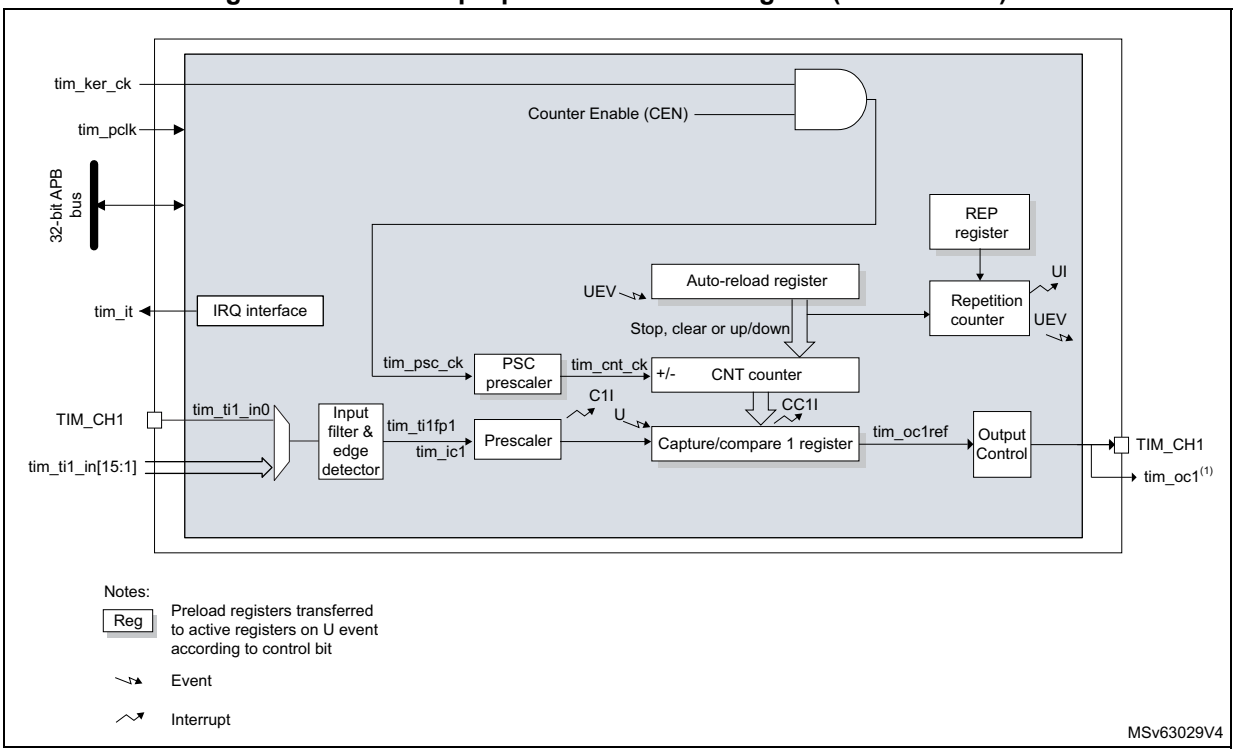


Figure 530. General-purpose timer block diagram (TIM13/TIM14)(2)



1. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 41.4.21: Using timer output as trigger for other timers (TIM13/TIM14 only)* for details.
2. Not available in STM32H523/533xx devices.

41.4.2 TIM12/TIM13/TIM14 pins and internal signals

Table 438 and *Table 439* in this section summarize the TIM inputs and outputs.

Table 438. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.

1. Available for TIM12 only.

Table 439. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).

Table 439. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_oc1 tim_oc2(1)	Output	Internal timer output. Can be used for triggering other timers or the ADC(s).
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests

1. Available for TIM12 only.

Table 440 and *Table 441* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 440. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM12	TIM13(1)	TIM14(1)
tim_ti1_in0	TIM12_CH1	TIM13_CH1	TIM14_CH1
tim_ti1_in1	Reserved	Reserved	
tim_ti1_in2			
tim_ti1_in3			
tim_ti1_in4	HSI/1024		
tim_ti1_in5	CSI/128		
tim_ti1_in[15:6]	Reserved		

1. Not available in STM32H523/533 devices.

Table 441. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM12
tim_ti2_in0	TIM12_CH2
tim_ti2_in[15:1]	Reserved

Table 442 lists the internal sources connected to the tim_itr input multiplexer.

Table 442. TIMx internal trigger connection

TIMx	TIM12
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	Reserved
tim_itr7	tim13_oc1(1)
tim_itr8	tim14_oc1(1)
tim_itr9	tim15_trgo
tim_itr10	tim16_oc1(1)
tim_itr11	tim17_oc1(1)
tim_itr[15:12]	Reserved

1. If the corresponding timer is not present, consider it as reserved.

41.4.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 531 and *Figure 532* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 531. Counter timing diagram with prescaler division change from 1 to 2

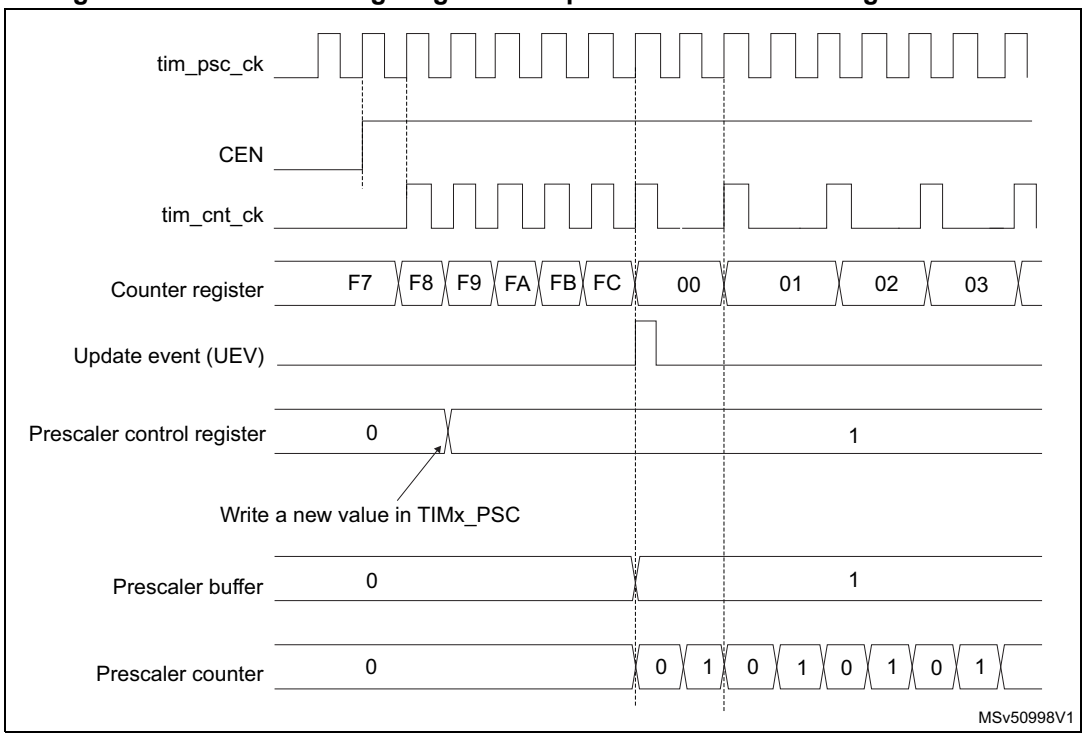
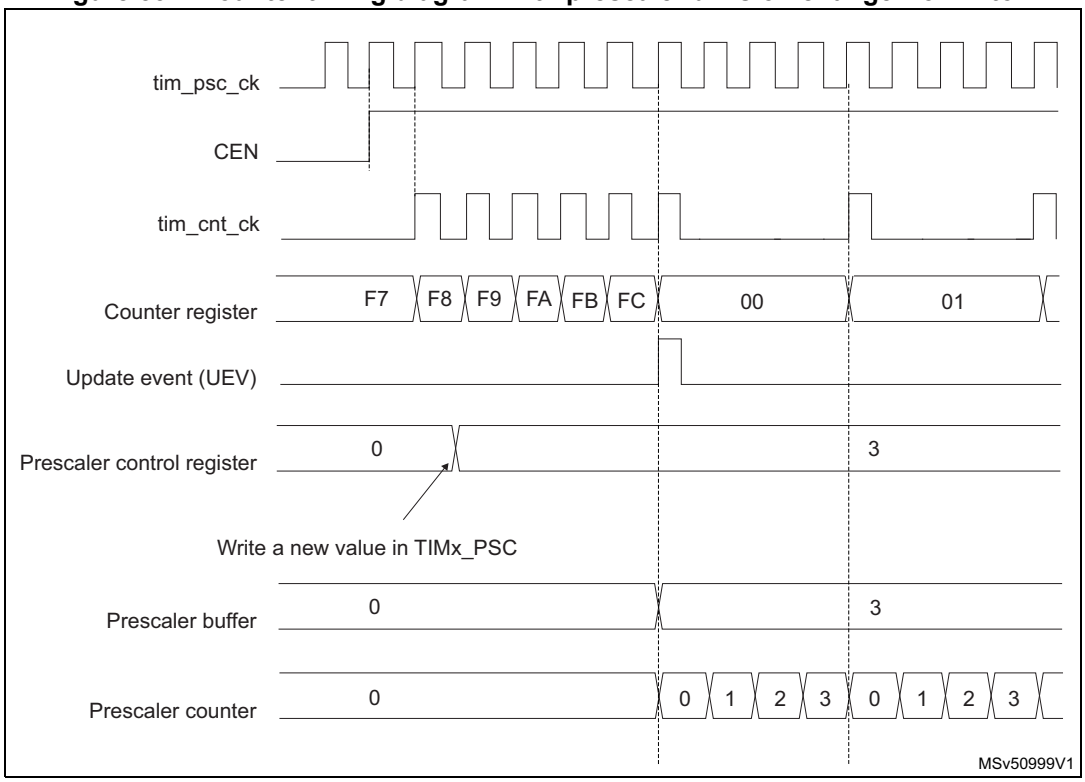


Figure 532. Counter timing diagram with prescaler division change from 1 to 4



41.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The autoreload shadow register is updated with the preload value (TIMx_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 533. Counter timing diagram, internal clock divided by 1

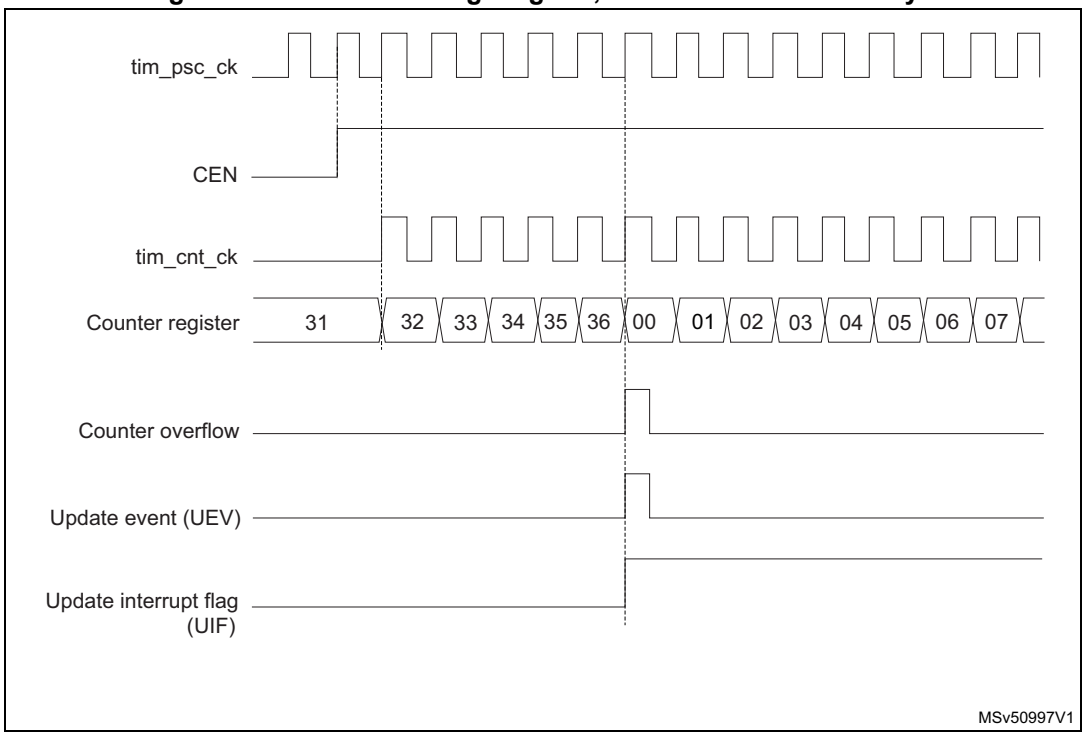


Figure 534. Counter timing diagram, internal clock divided by 2

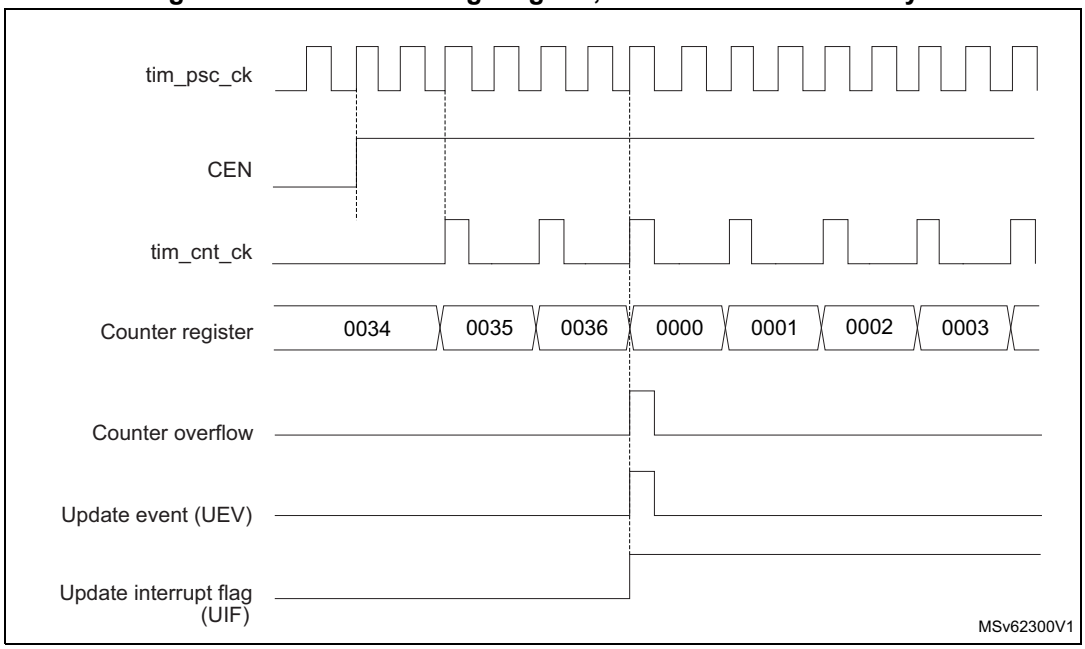


Figure 535. Counter timing diagram, internal clock divided by 4

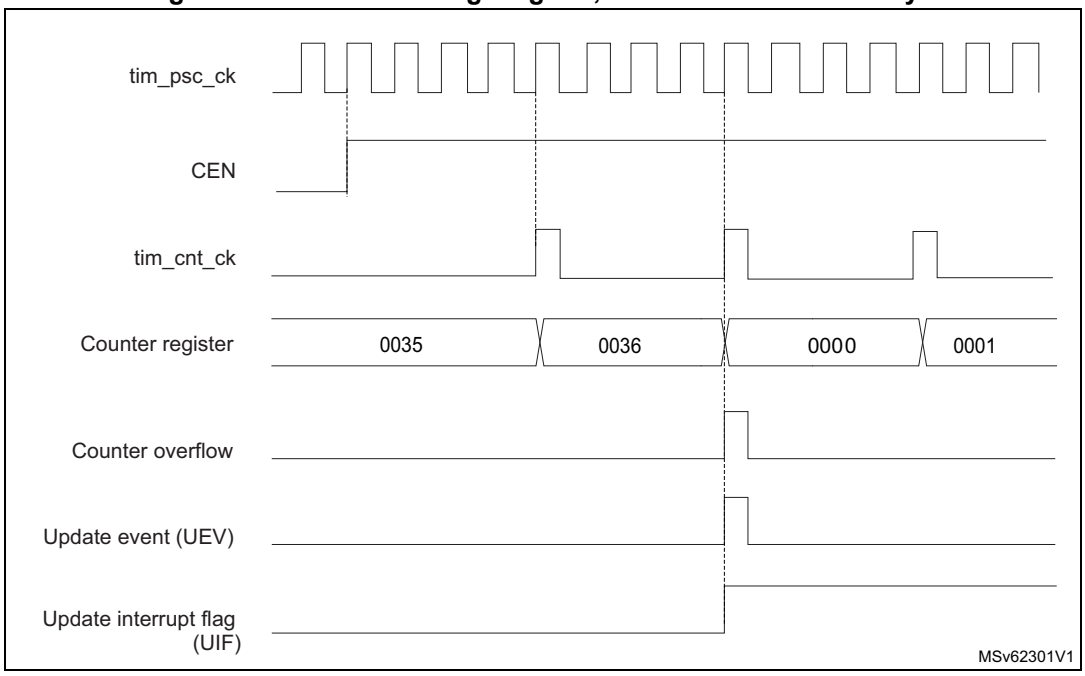


Figure 536. Counter timing diagram, internal clock divided by N

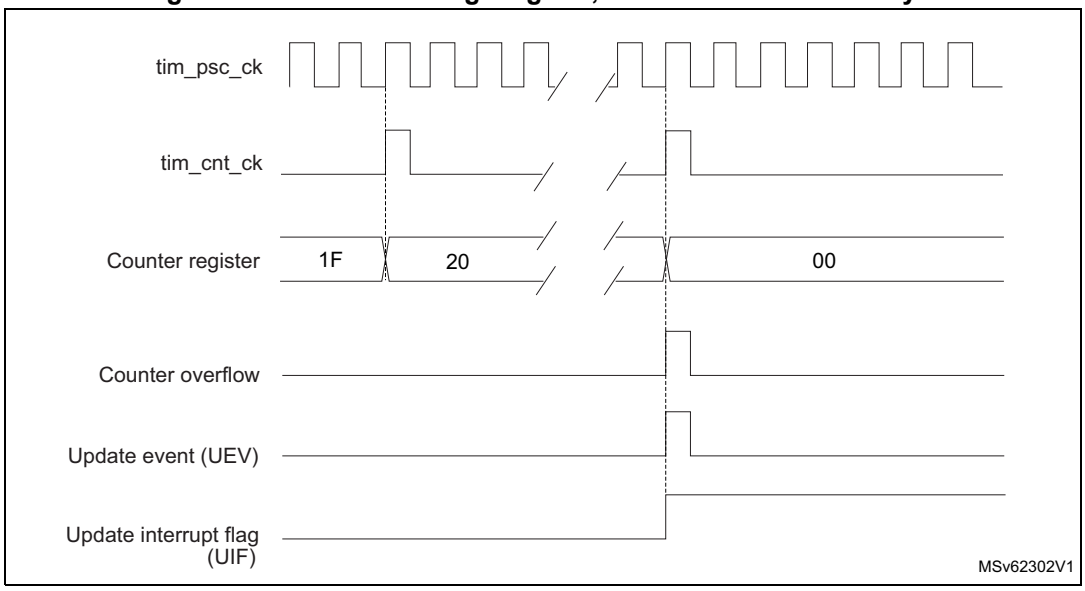


Figure 537. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

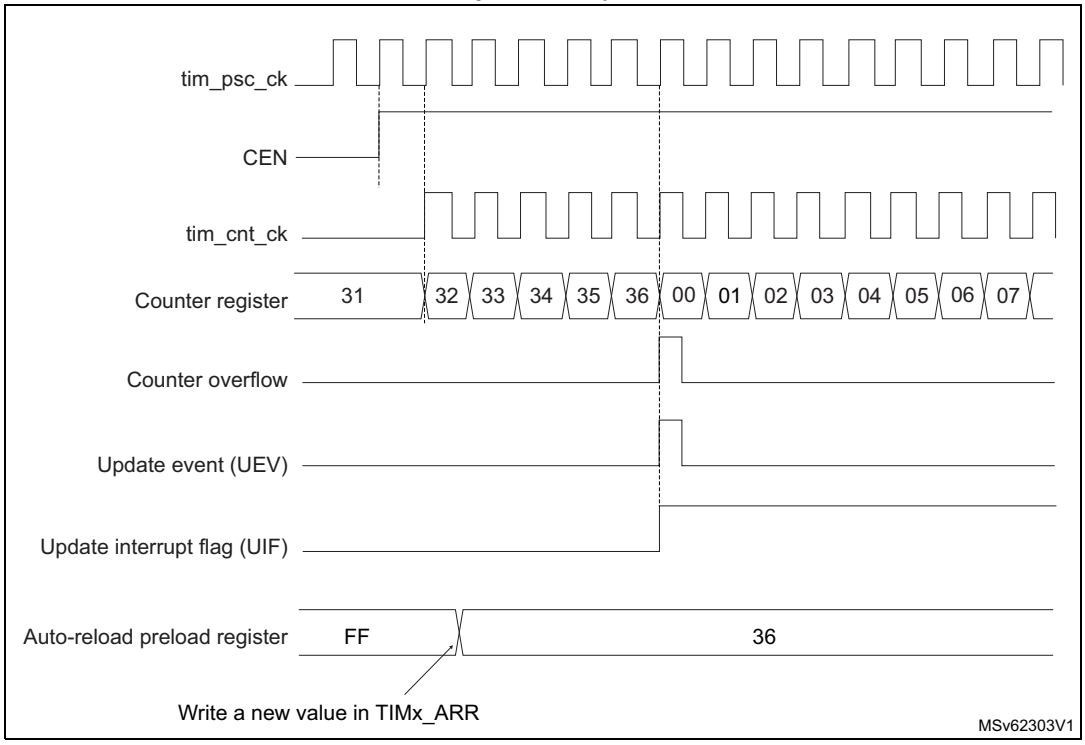
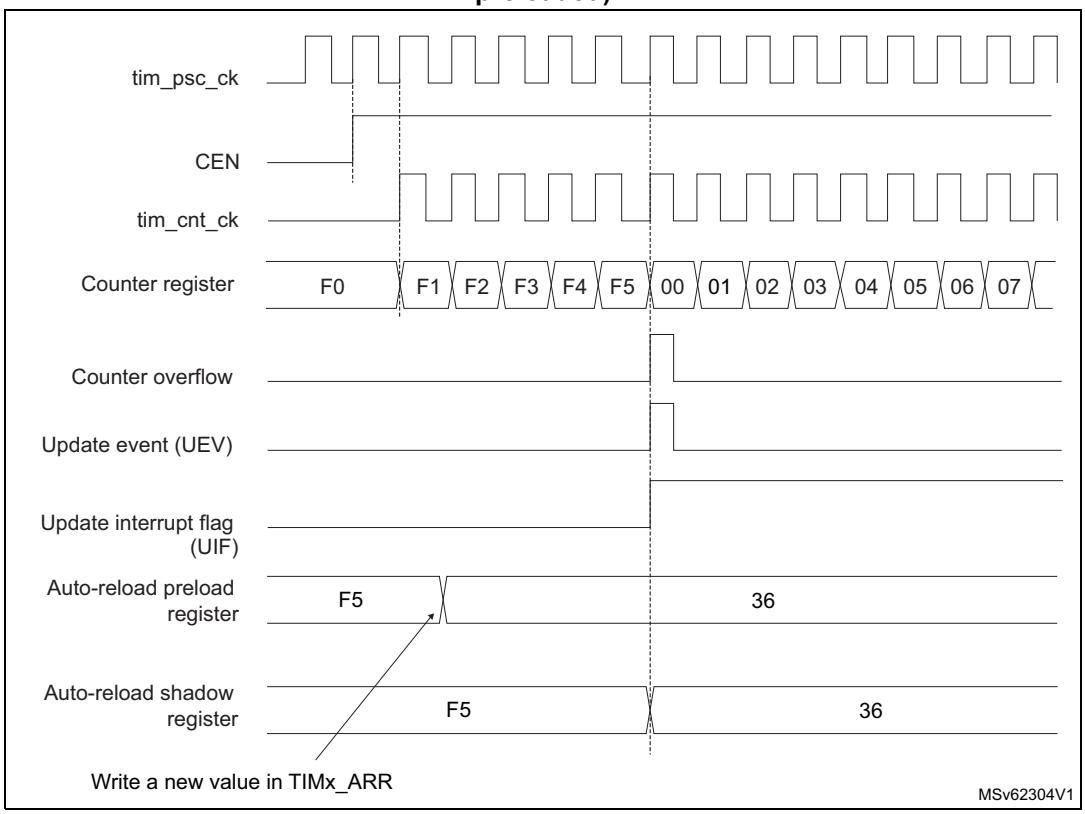


Figure 538. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



41.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1 (for TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (for TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

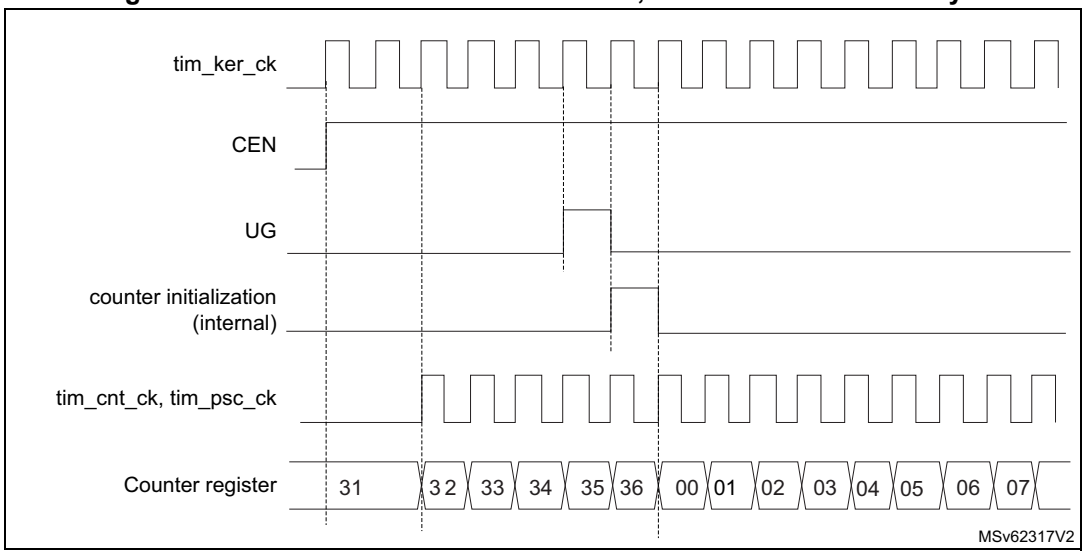
Internal clock source (tim_ker_ck)

The internal clock source is the default clock source for TIM13/TIM14.

For TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS = 000). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 539 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

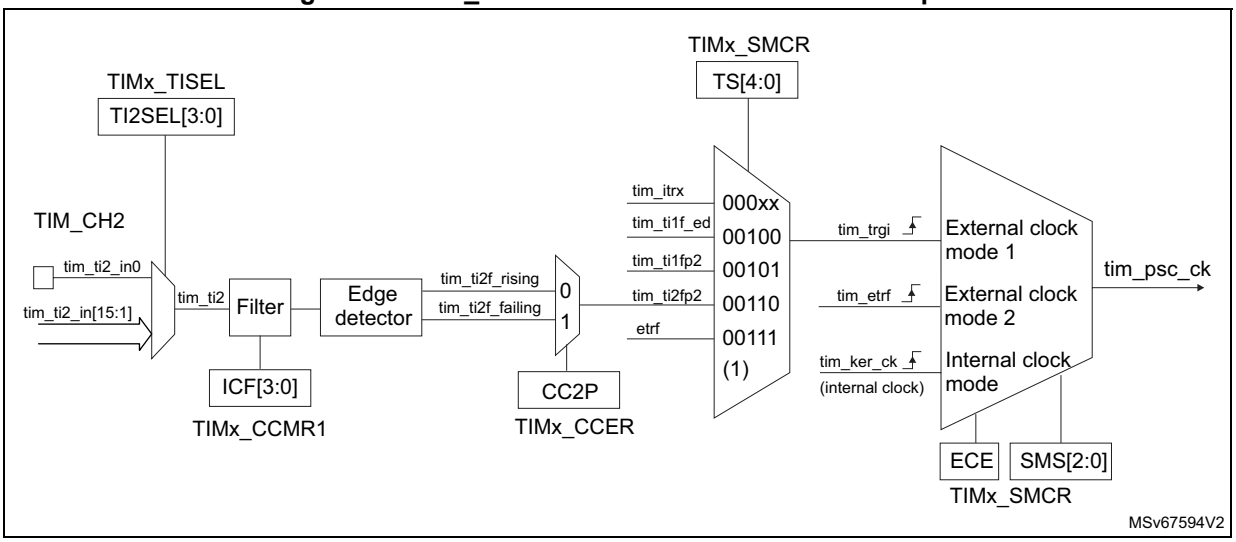
Figure 539. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 540. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

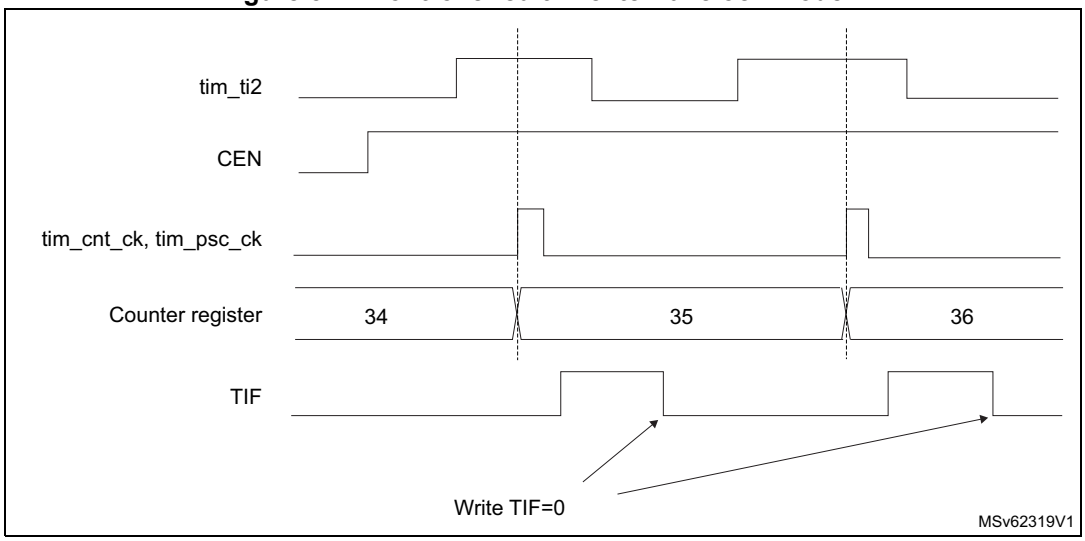
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select the rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 541. Control circuit in external clock mode 1



41.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler), and an output stage (with comparator and output control).

Figure 542, *Figure 543*, *Figure 544* and *Figure 545* give an overview of a capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which

can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 542. Capture/compare channel 1 input stage (TIM13/TIM14)

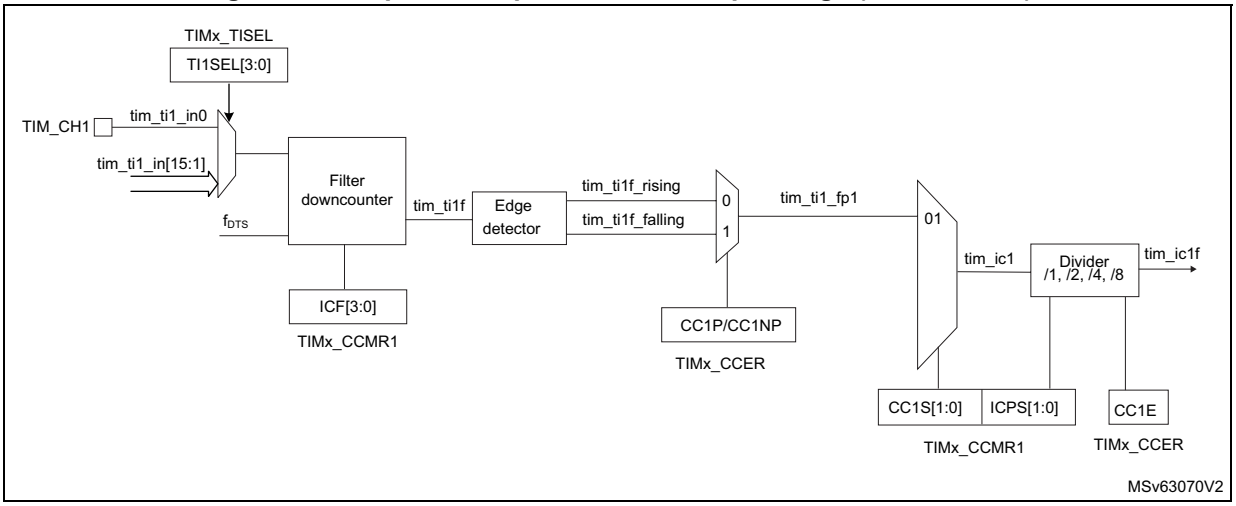
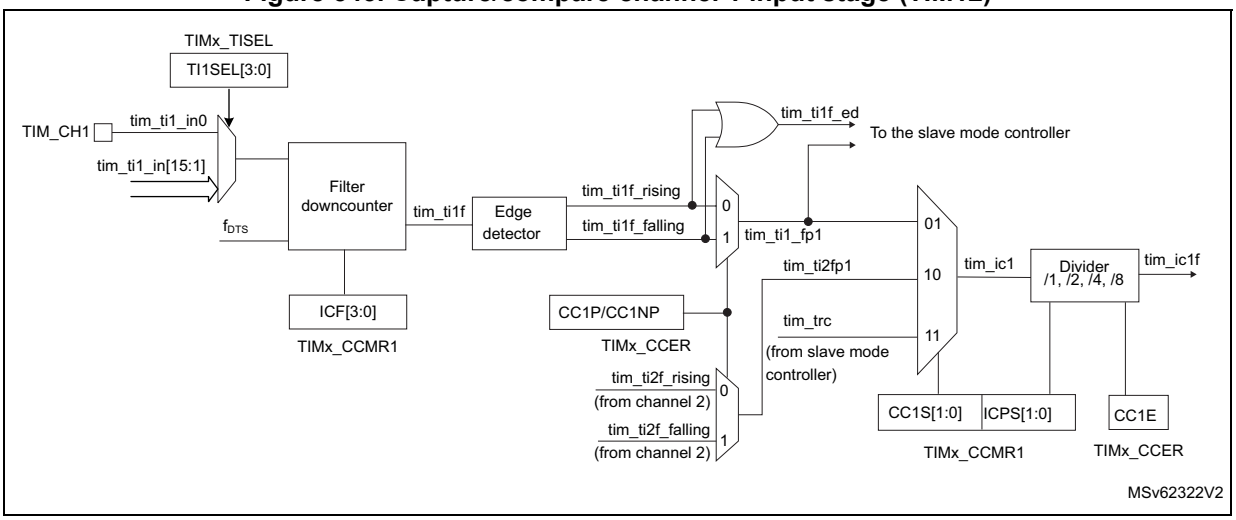


Figure 543. Capture/compare channel 1 input stage (TIM12)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 544. Capture/compare channel 1 main circuit

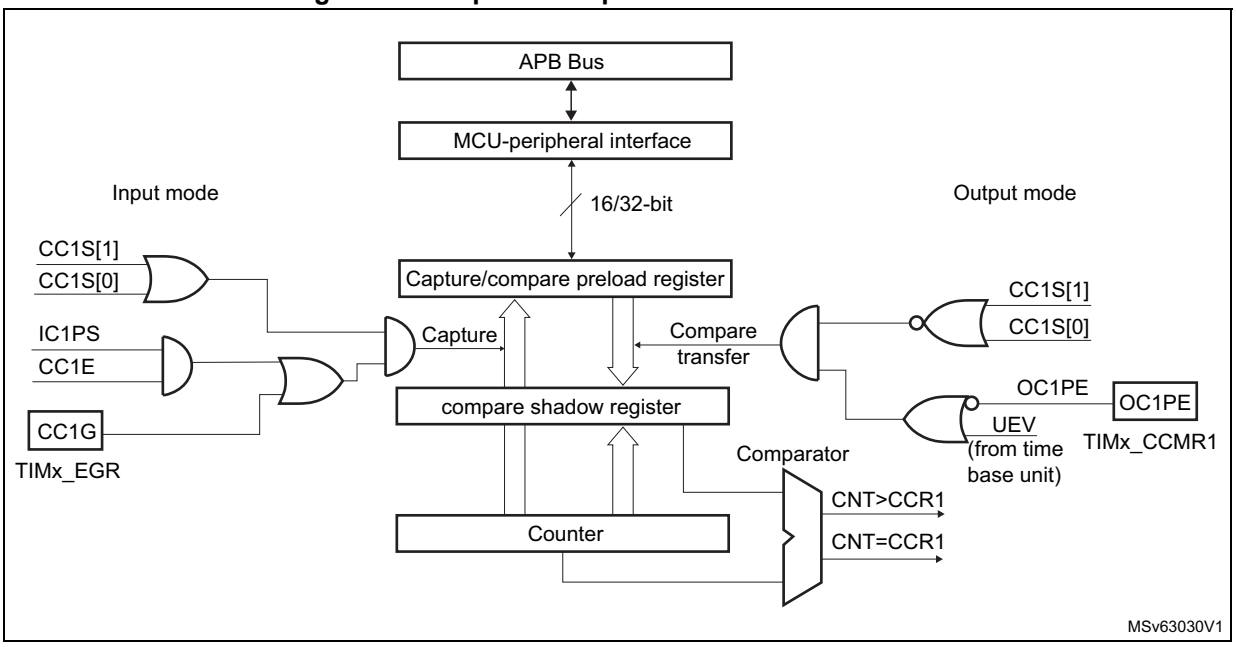
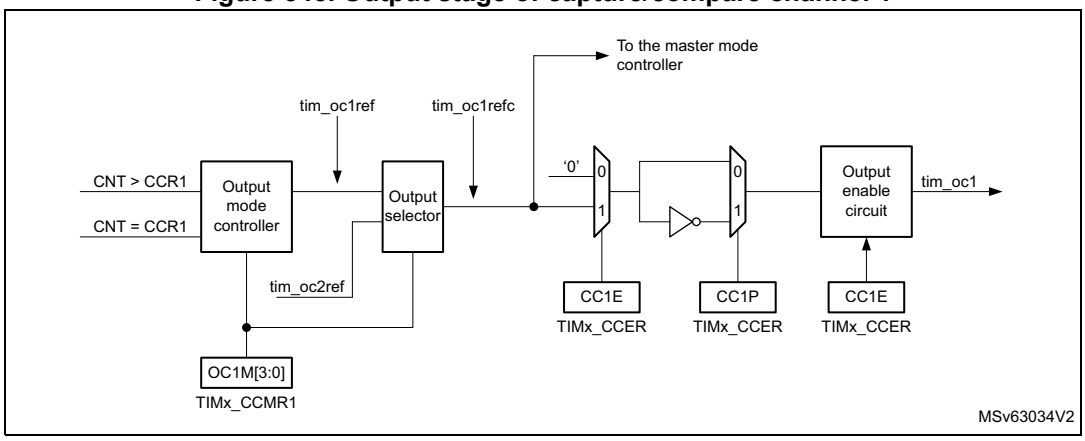


Figure 545. Output stage of capture/compare channel 1



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

41.4.7 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be

cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input mode, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the tim_tix inputs). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by programming CC1P and CC1NP bits to 00 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wishes the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

41.4.8 PWM input mode (TIM12 only)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges.

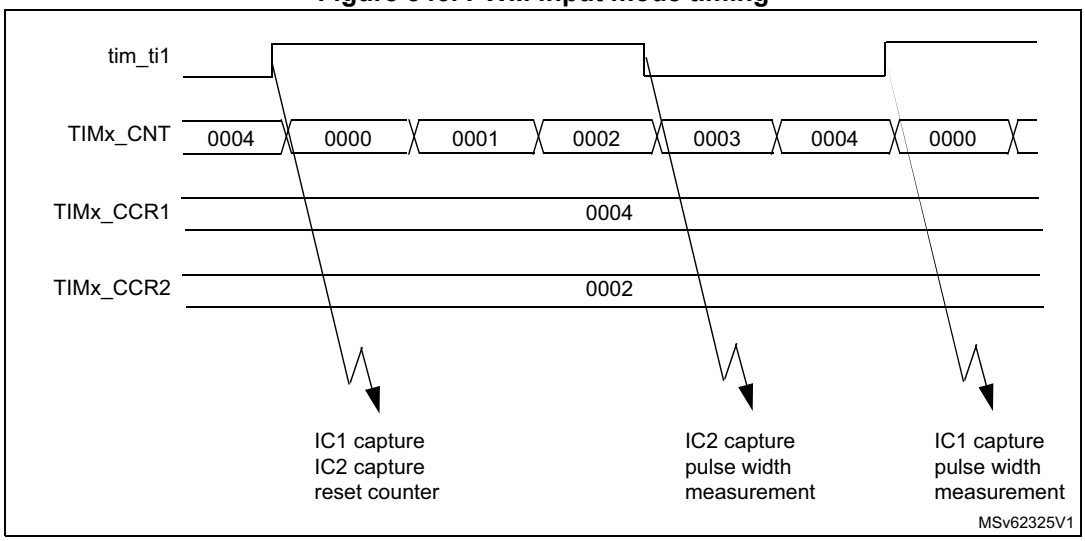
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to 00 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 546. PWM input mode timing



41.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

41.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

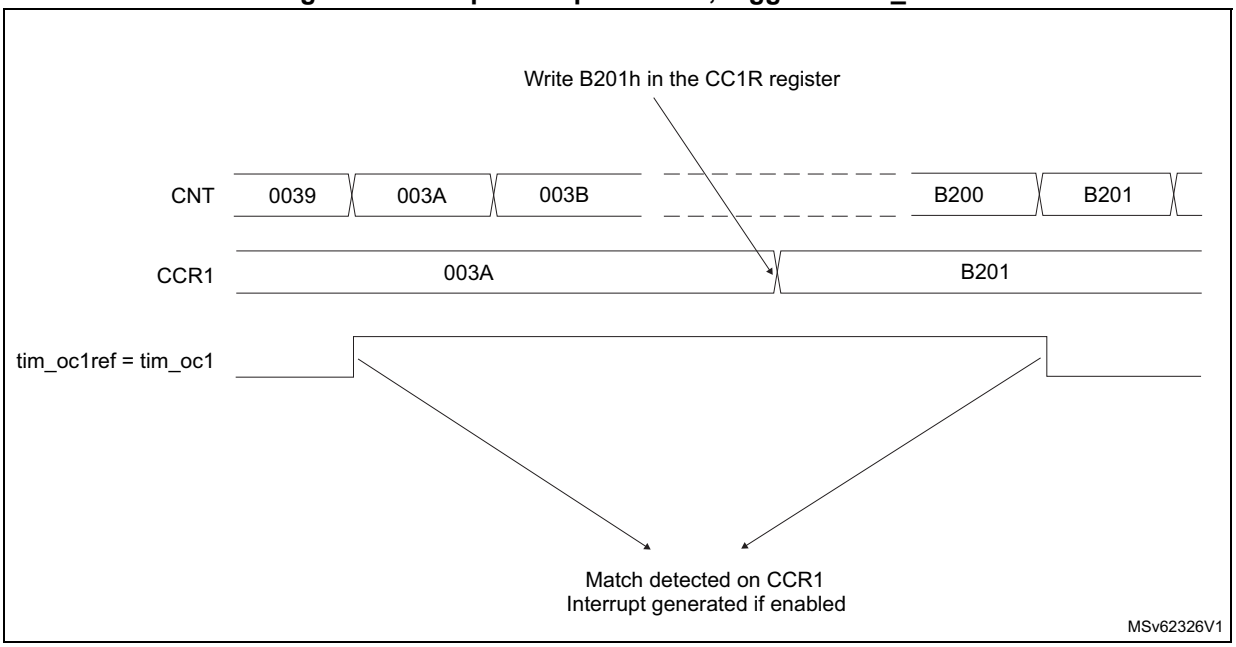
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 547*.

Figure 547. Output compare mode, toggle on tim_oc1.



41.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

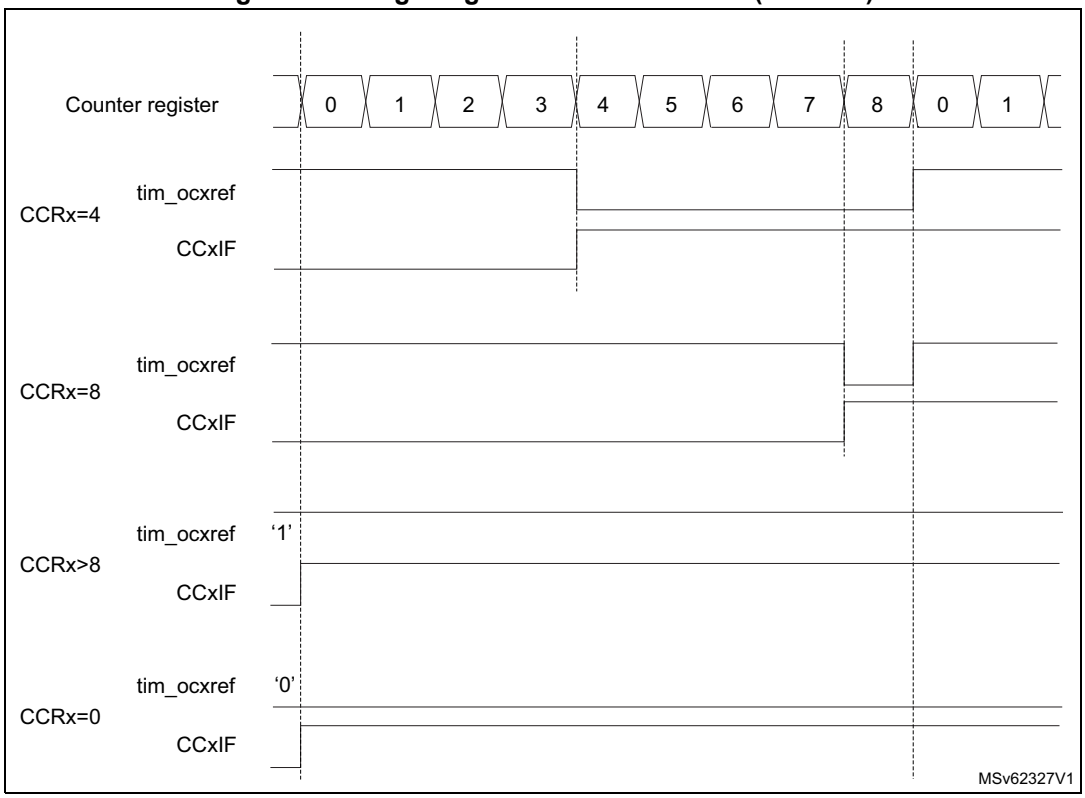
The tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, the user considers PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 548* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 548. Edge-aligned PWM waveforms (ARR = 8)

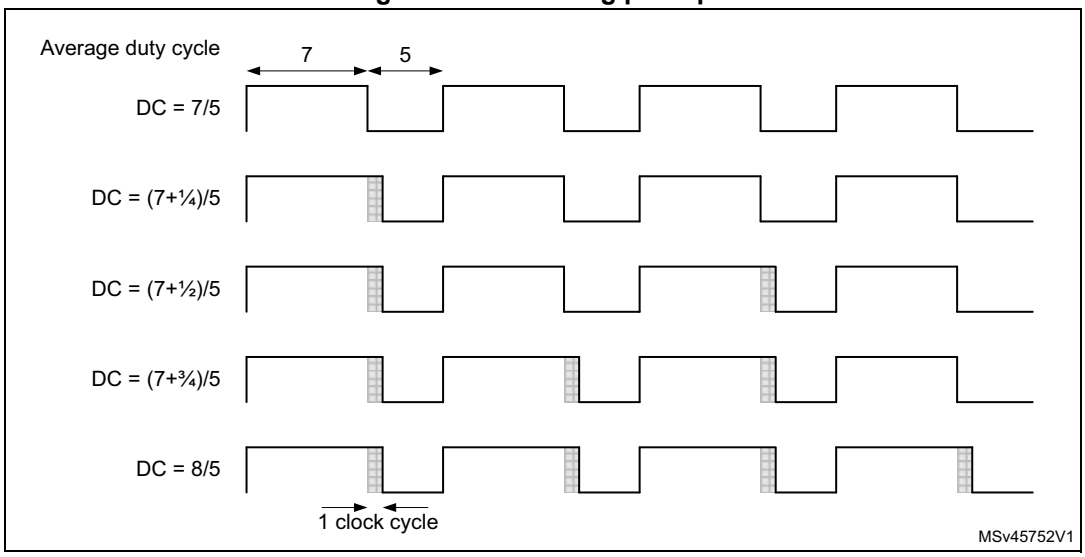


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 549* presents the dithering principle applied to four consecutive PWM cycles.

Figure 549. Dithering principle

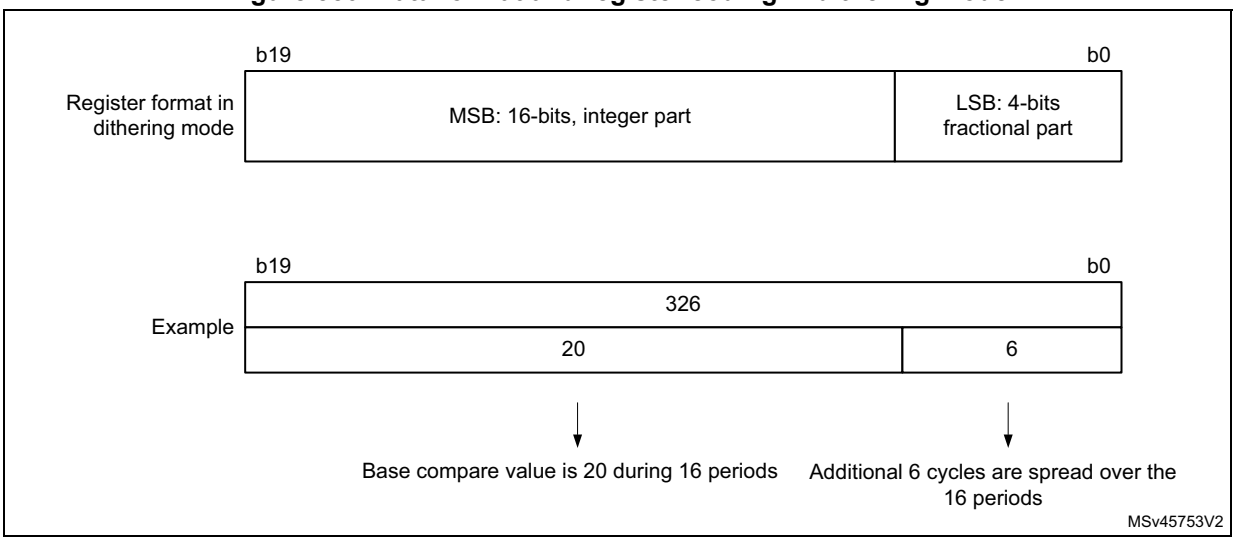


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 550* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 550. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

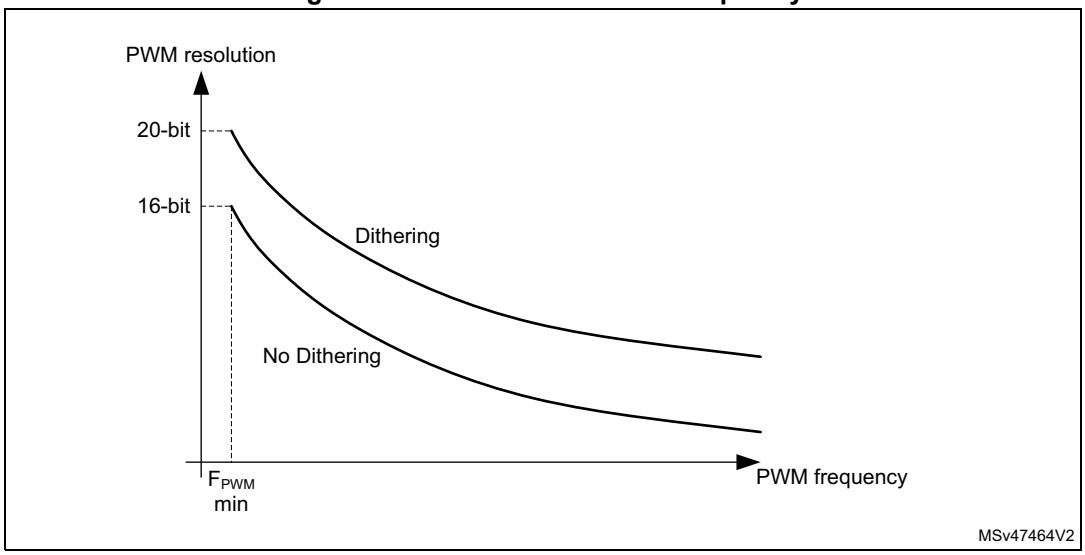
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

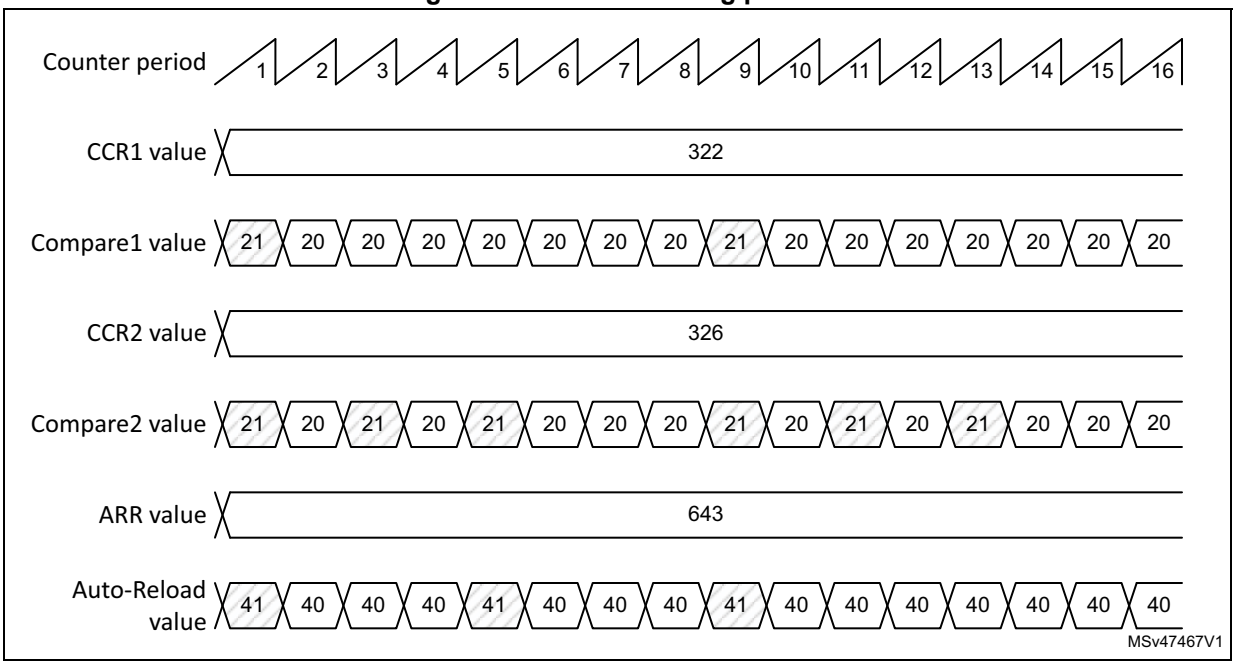
As shown on *Figure 551*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 551. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 552*.

Figure 552. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 443*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 443. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 443. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

41.4.12 Combined PWM mode (TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

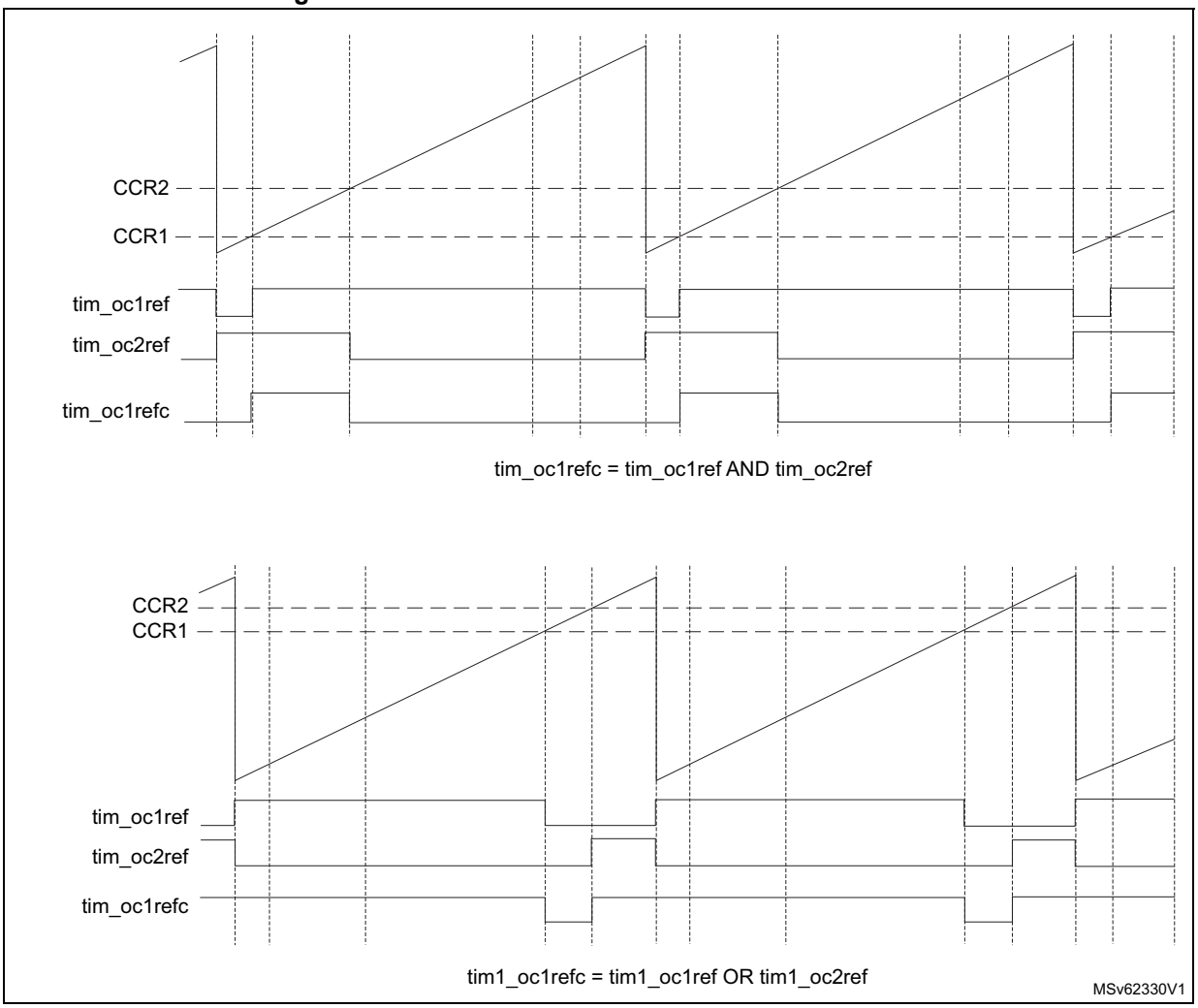
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 553 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 553. Combined PWM mode on channel 1 and 2



41.4.13 One-pulse mode

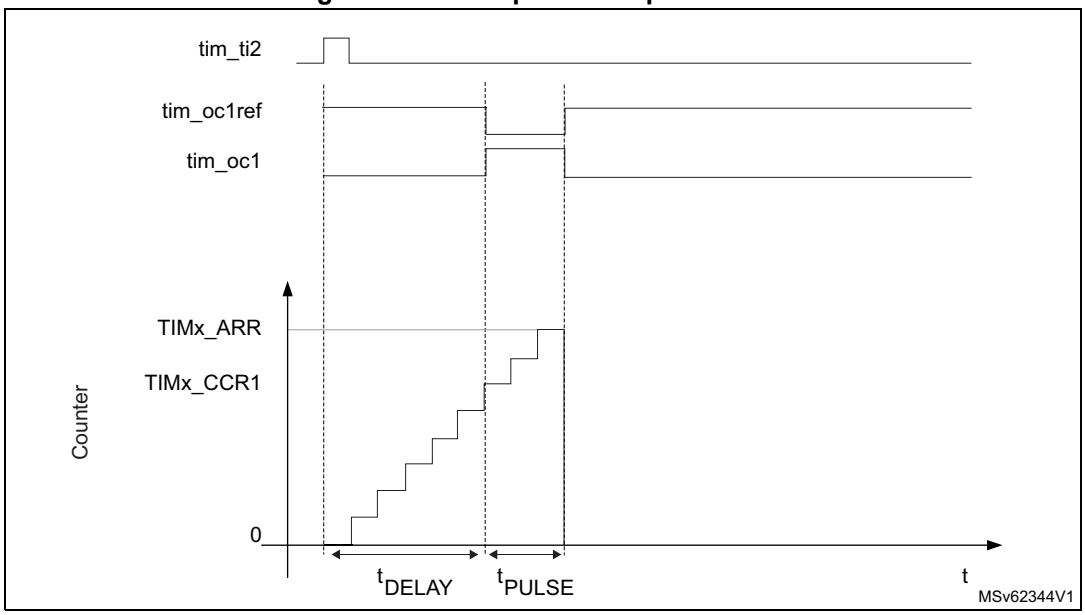
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 554. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Assuming the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 0111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

41.4.14 Retriggerable one pulse mode (TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one pulse mode described in *Section 41.4.13: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

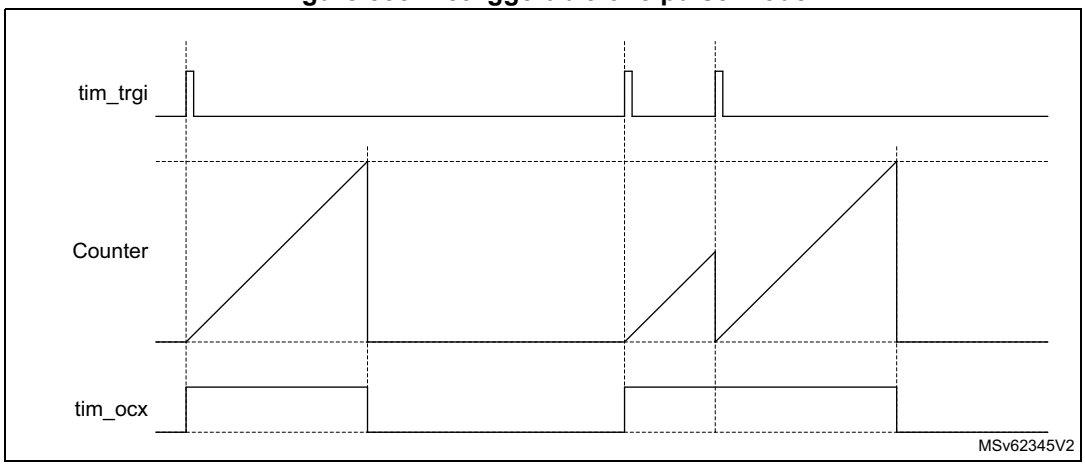
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 555. Retriggerable one pulse mode



41.4.15 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

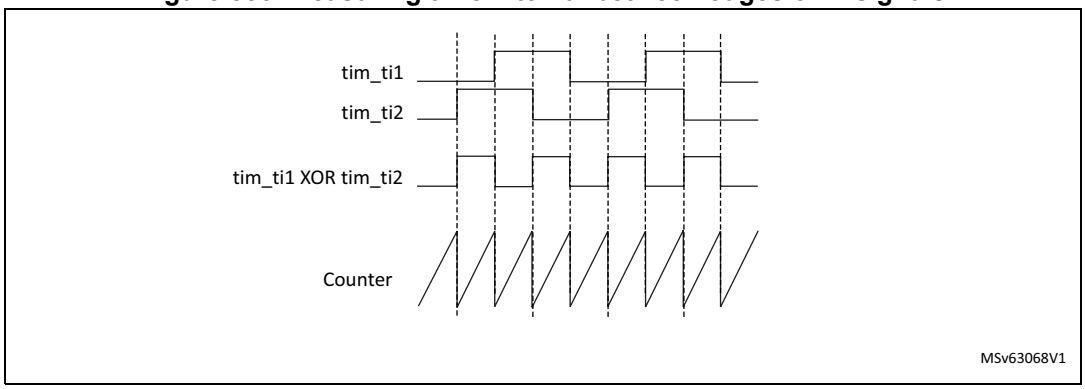
There is no latency between the assertions of the UIF and UIFCPY flags.

41.4.16 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 556*.

Figure 556. Measuring time interval between edges on 2 signals



41.4.17 TIM12 external trigger synchronization

The TIM12 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and Gated + reset mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the

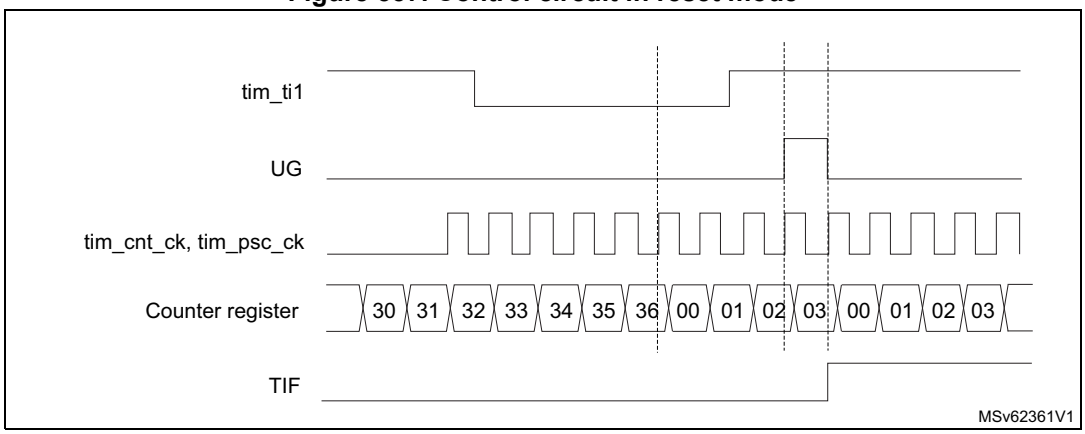
TIMx_CCMR1 register. Program CC1P and CC1NP to 00 in TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 557. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

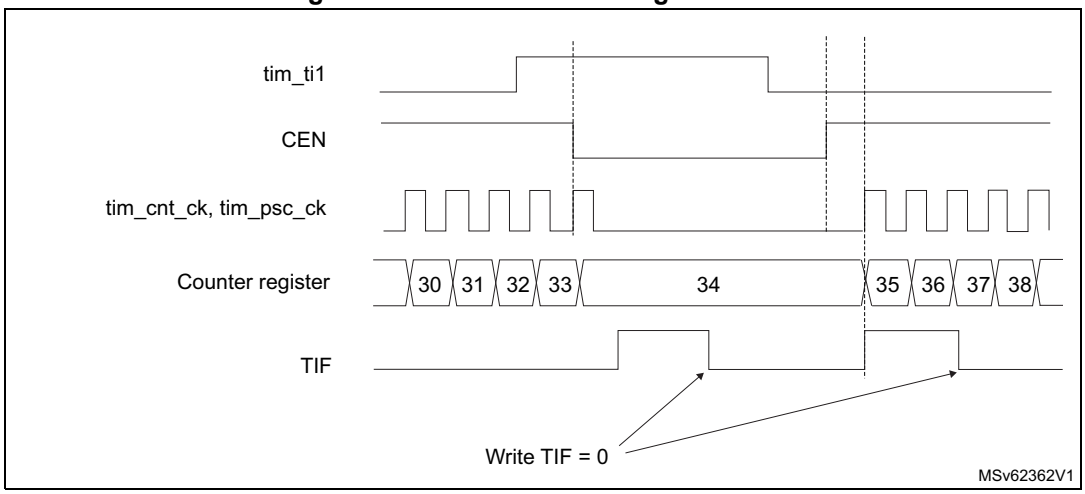
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 558. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

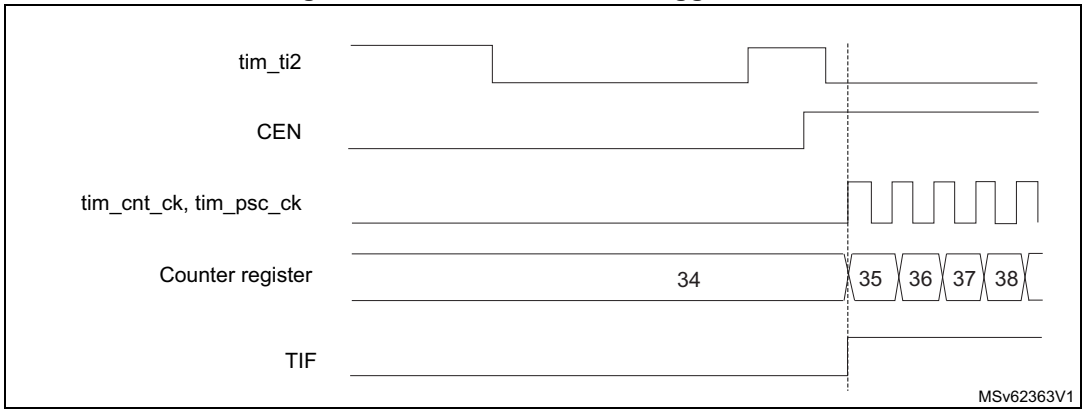
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Program CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 559. Control circuit in trigger mode



41.4.18 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

41.4.19 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

41.4.20 Timer synchronization (TIM12 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 39.4.23: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

41.4.21 Using timer output as trigger for other timers (TIM13/TIM14 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

41.4.22 ADC triggers (TIM12 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

41.4.23 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the debug section.

41.5 TIM12/TIM13/TIM14 low-power modes

Table 444. Effect of low-power modes on TIM12/TIM13/TIM14

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

41.6 TIM12/TIM13/TIM14 interrupts

The TIM12/TIM13/TIM14 can generate multiple interrupts, as shown in *Table 445*.

Table 445. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
TIM	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No

1. Available for TIM12 only.

41.7 TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

41.7.1 TIM12 control register 1 (TIM12_CR1)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled. These events can be:
– Counter overflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit
Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
CEN is cleared automatically in one-pulse mode, when an update event occurs.
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

41.7.2 TIM12 control register 2 (TIM12_CR2)

Address offset: 0x004

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: Update - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: Compare - tim_oc1refc signal is used as trigger output (tim_trgo).

101: Compare - tim_oc2refc signal is used as trigger output (tim_trgo).

Bits 3:0 Reserved, must be kept at reset value.

41.7.3 TIM12 slave mode control register (TIM12_SMCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw

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

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

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal Trigger 10 (tim_itr11)
10000: Internal Trigger 10 (tim_itr12)
10001: Internal Trigger 10 (tim_itr13)
10010: Internal Trigger 10 (tim_itr14)
10011: Internal Trigger 10 (tim_itr15)
Others: Reserved

See *Table 442: TIMx internal trigger connection* for more details on the meaning of tim_itrx for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see input control register and control register description.

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Other codes: reserved.

Note: The gated mode (including gated + reset mode) must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

41.7.4 TIM12 interrupt enable register (TIM12_DIER)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

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

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.

1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

41.7.5 TIM12 status register (TIM12_SR)

Address offset: 0x010

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow and if UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 41.7.3: TIM12 slave mode control register (TIM12_SMCR)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

41.7.6 TIM12 event generation register (TIM12_EGR)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

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

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

41.7.7 TIM12 capture/compare mode register 1 (TIM12_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

41.7.8 TIM12 capture/compare mode register 1 [alternate] (TIM12_CCMR1)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas the active level of tim_oc1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The tim_oc1ref signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. The tim_oc1ref signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1

0100: Force inactive level - tim_oc1ref is forced low

0101: Force active level - tim_oc1ref is forced high

0110: PWM mode 1 - channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active

1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00:CC1 channel is configured as output

01:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11:CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

41.7.9 TIM12 capture/compare enable register (TIM12_CCER)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity

Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity

Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable

Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity

CC1 channel configured as output: CC1NP must be kept cleared

CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active

1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 446. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The states of the external I/O pins connected to the standard tim_ocx *channels depend on the state of the* tim_ocx *channel and on the GPIO registers.*

41.7.10 TIM12 counter (TIM12_CNT)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

41.7.11 TIM12 prescaler (TIM12_PSC)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

41.7.12 TIM12 autoreload register (TIM12_ARR)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 41.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

41.7.13 TIM12 capture/compare register 1 (TIM12_CCR1)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

41.7.14 TIM12 capture/compare register 2 (TIM12_CCR2)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

41.7.15 TIM12 timer input selection register (TIM12_TISEL)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

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

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIMx_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

0100: tim_ti2_in15

Refer to *Table 441: Interconnect to the tim_ti2 input multiplexer* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIMx_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Table 440: Interconnect to the tim_ti1 input multiplexer* for interconnects list.

41.7.16 TIM12 register map

TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 447. TIM12 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIM12_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0				0	0	0	0
0x004	**TIM12_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x008	**TIM12_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x00C	**TIM12_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																										0				0	0	0
0x010	**TIM12_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x014	**TIM12_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x018	**TIM12_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM12_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	**TIM12_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x024	**TIM12_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	**TIM12_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 447. TIM12 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	**TIM12_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	**TIM12_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	**TIM12_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C to 0x058	Reserved	Res.																															
0x05C	**TIM12_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

41.8 TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

41.8.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

– Counter overflow

– Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

– Counter overflow

– Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

41.8.2 TIMx interrupt enable register (TIMx_DIER)(x = 13, 14)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

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

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

41.8.3 TIMx status register (TIMx_SR)(x = 13, 14)

Address offset: 0x010

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow and if UDIS = 0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

41.8.4 TIMx event generation register (TIMx_EGR)(x = 13, 14)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w

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

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

41.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13, 14)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

41.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13, 14)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

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

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active

Others: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: Reserved.

11: Reserved.

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

41.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 13, 14)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define tim_ti1fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 for capture operations.
CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_ti1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_ti1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_ti1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 not inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 448. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output Disabled (tim_ocx = 0)
1	tim_ocx = tim_ocxref + Polarity

Note: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.

41.8.8 TIMx counter (TIMx_CNT)(x = 13, 14)

Address offset: 0x024

Reset value: 0x0000 0000

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

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

41.8.9 TIMx prescaler (TIMx_PSC)(x = 13, 14)

Address offset: 0x028

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

41.8.10 TIMx autoreload register (TIMx_ARR)(x = 13, 14)

Address offset: 0x02C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 41.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

41.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13, 14)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

41.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 13, 14)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Table 440: Interconnect to the tim_ti1 input multiplexer* for interconnects list.

41.8.13 TIM13/TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 449. TIM13/TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**TIMx_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0				0	0	0	0
0x004 to 0x008	Reserved	Res.																															
0x00C	**TIMx_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
	Reset value																															0	0
0x010	**TIMx_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0								0	0
0x014	**TIMx_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																															0	0
0x018	**TIMx_CCMR1** Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	**TIMx_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
	Reset value																													0		0	0
0x024	**TIMx_CNT**	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 449. TIM13/TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	**TIMx_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	**TIMx_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	**TIMx_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x058	Reserved	Res.																															
0x05C	**TIMx_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

42 General purpose timers (TIM15/TIM16/TIM17)

42.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 42.4.26: Timer synchronization (TIM15 only)*.

42.2 TIM15 main features

TIM15 includes the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)

42.3 TIM16/TIM17 main features

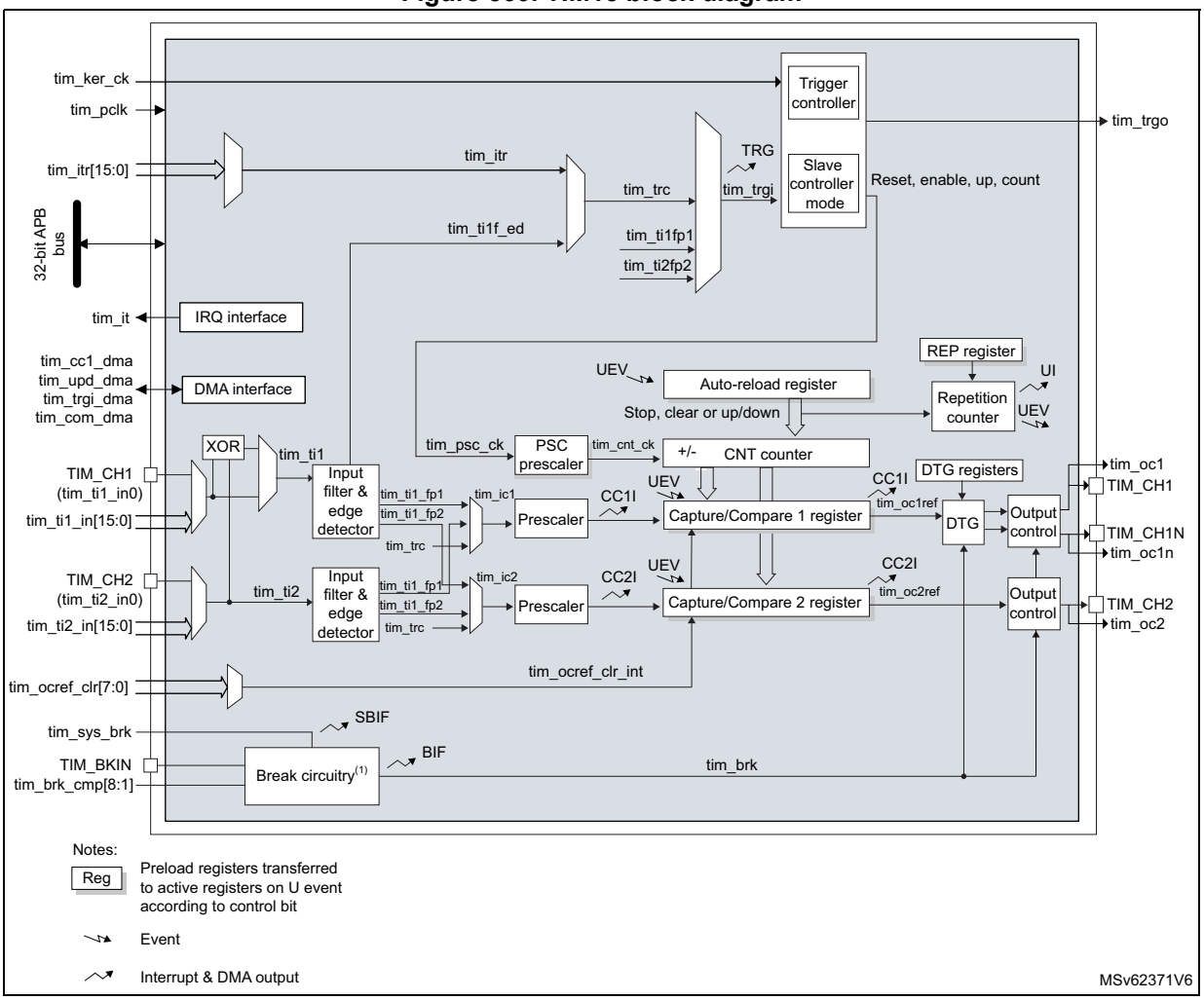
The TIM16/TIM17 timers include the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

42.4 TIM15/TIM16/TIM17 functional description

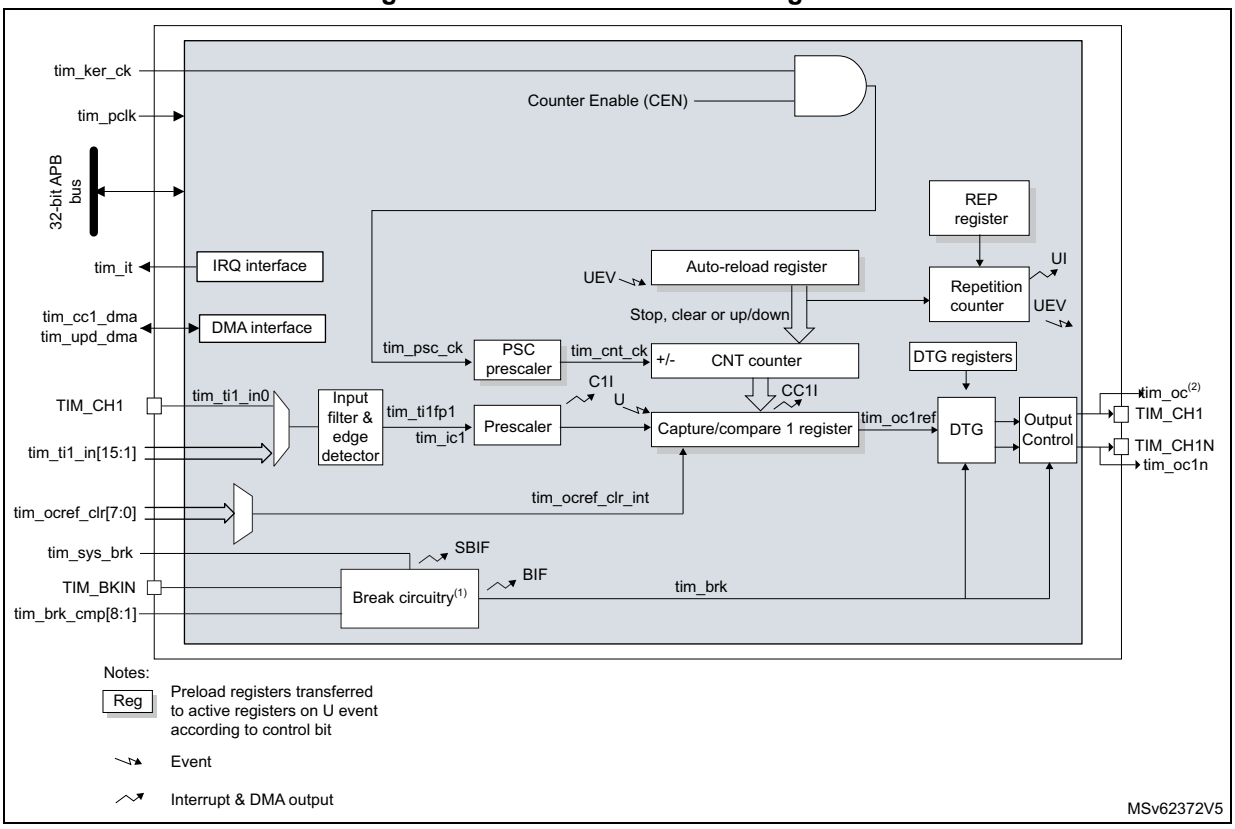
42.4.1 Block diagram

Figure 560. TIM15 block diagram



1. Refer to *Section 42.4.15: Using the break function* for details.

Figure 561. TIM16/TIM17 block diagram(3)



1. Refer to *Section 42.4.15: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 42.4.27: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.
3. Not available in STM32H523/533xx devices.

42.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 450 and *Table 451* in this section summarize the TIM inputs and outputs.

Table 450. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 451. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

Table 452 and *Table 453* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 452. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16(1)	TIM17(1)
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	TIM2_CH1	RCC_LSI	Reserved
tim_ti1_in2	TIM3_CH1	RCC_LSE	RCC_HSE_1MHz(2)
tim_ti1_in3	TIM4_CH1	RTC Wake-up	RCC_MCO1
tim_ti1_in4	RCC_LSE	Reserved	Reserved
tim_ti1_in5	RCC_CSI/128		
tim_ti1_in6	RCC_MCO2		
tim_ti1_in[15:6]	Reserved		

1. Not available in STM32H523/533 devices.
2. This signal is available only if the RTCPRE bit is set in the TIM17_OR1 register. See *Section 42.8.19: TIM17 option register 1 (TIM17_OR1)* for details.

Table 453. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources
	TIM15
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	TIM2_CH2
tim_ti2_in2	TIM3_CH2
tim_ti2_in3	TIM4_CH2
tim_ti2_in[15:4]	Reserved

Table 454 lists the internal sources connected to the tim_itr input multiplexer.

Table 454. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	tim12_trgo
tim_itr7	tim13_oc1(1)

Table 454. TIMx internal trigger connection (continued)

tim_itrx inputs	TIM15
tim_itr8	tim14_oc1(1)
tim_itr9	Reserved
tim_itr10	tim16_oc1(1)
tim_itr11	tim17_oc1(1)
tim_itr[15:12]	Reserved

1. If corresponding timer is not present, consider it as reserved.

Table 455 and *Table 456* list the sources connected to the tim_brk input.

Table 455. Timer break interconnect

tim_brk inputs	TIM15	TIM16(1)	TIM17(1)
TIM_BKIN	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp[8:1]	Reserved		

1. Not available in STM32H523/533 devices.

Table 456. System break interconnect

tim_sys_brk inputs	TIM15/16/17	Enable bit in SBS_CFGR2 register
tim_sys_brk0	FLASH double ECC error	ECCL
tim_sys_brk1	Programmable voltage detector (PVD)	PVDL
tim_sys_brk2	SRAM double ECC error	SEL
tim_sys_brk3	Cortex-M33 LOCKUP	CLL
CSS	Clock security system	None (always enabled)

42.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload

preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Table 572 and *Table 563* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 562. Counter timing diagram with prescaler division change from 1 to 2

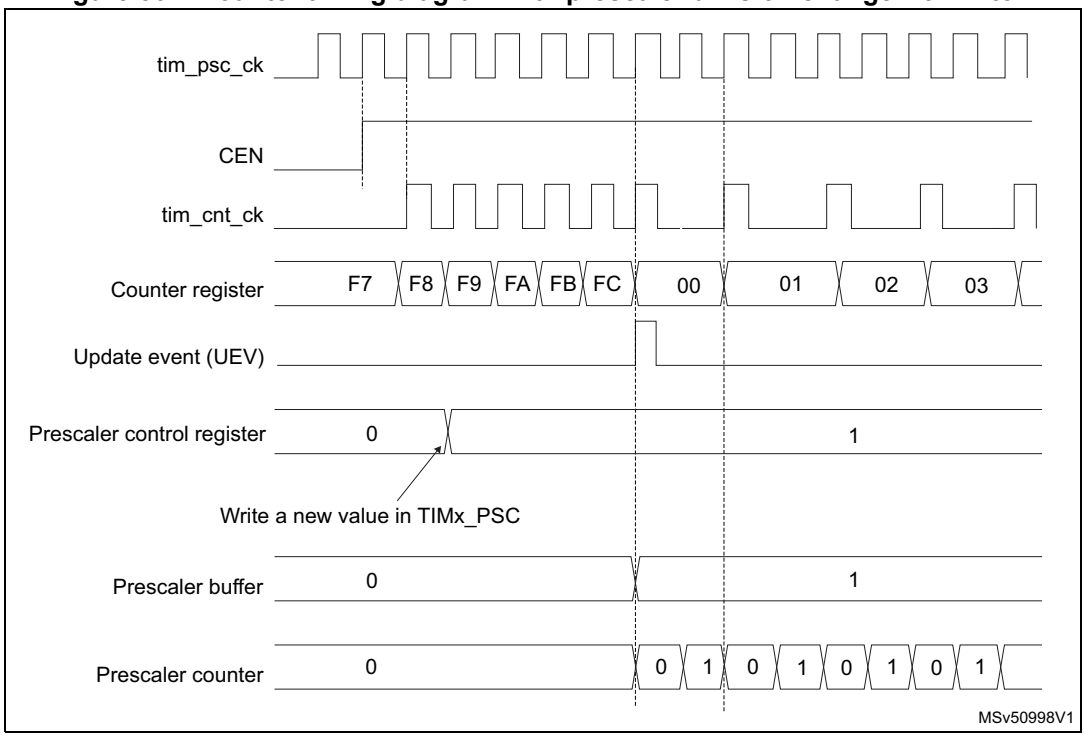
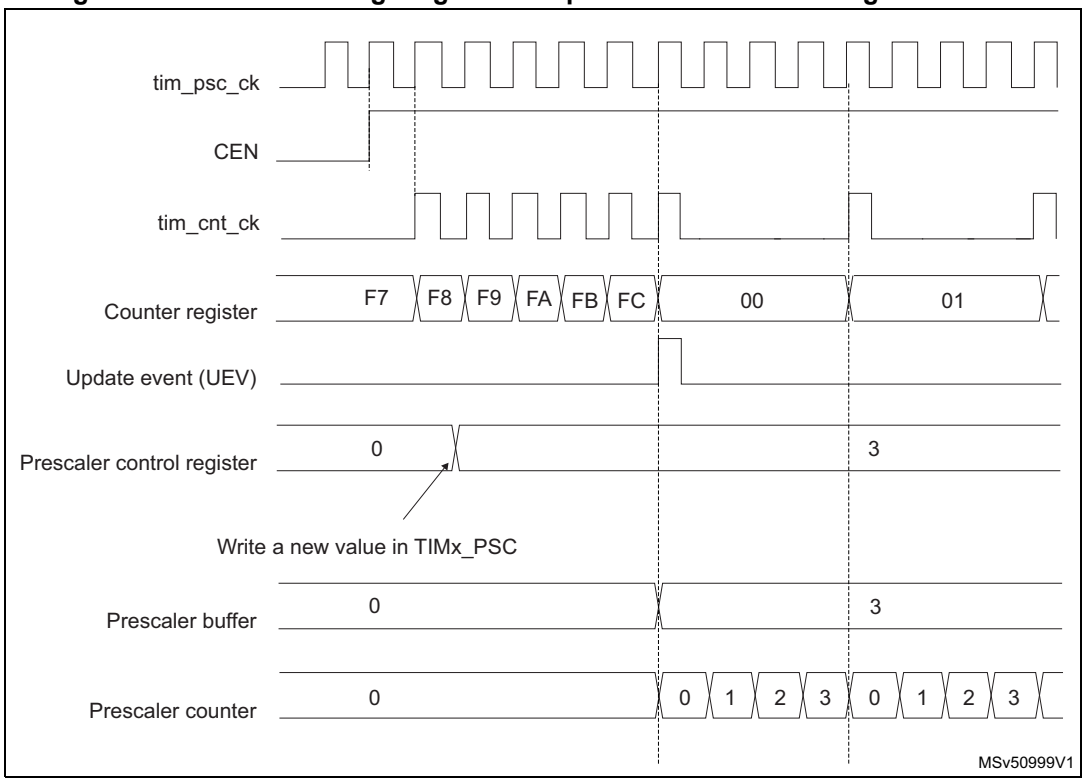


Figure 563. Counter timing diagram with prescaler division change from 1 to 4



42.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 564. Counter timing diagram, internal clock divided by 1

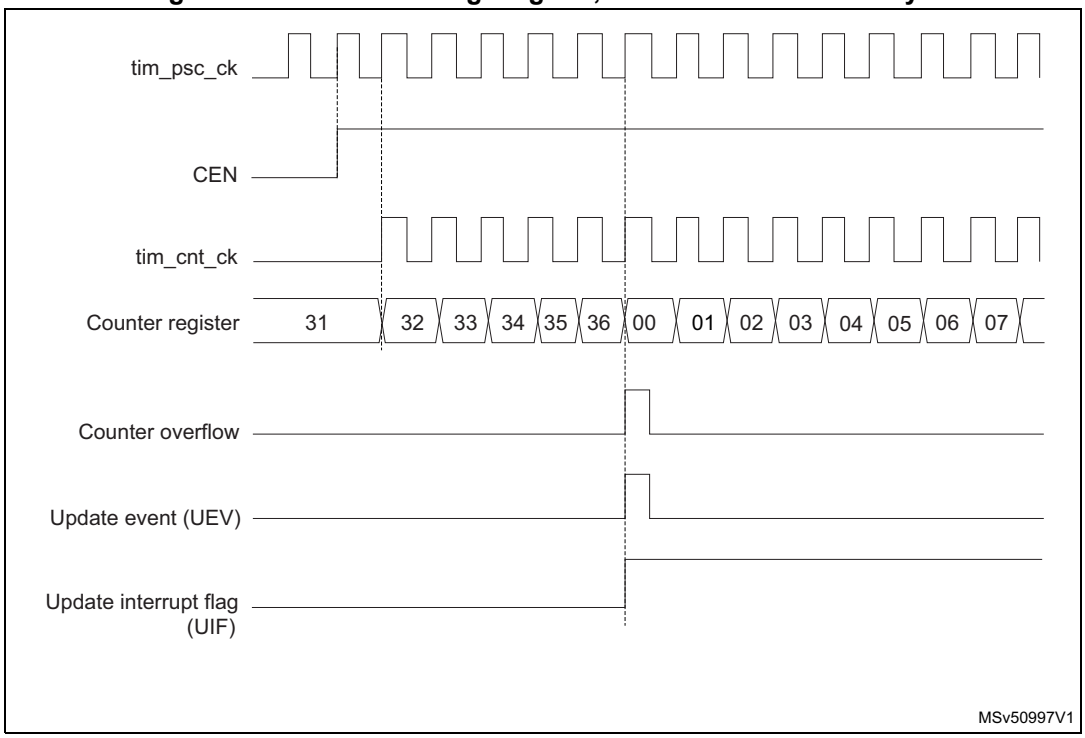


Figure 565. Counter timing diagram, internal clock divided by 2

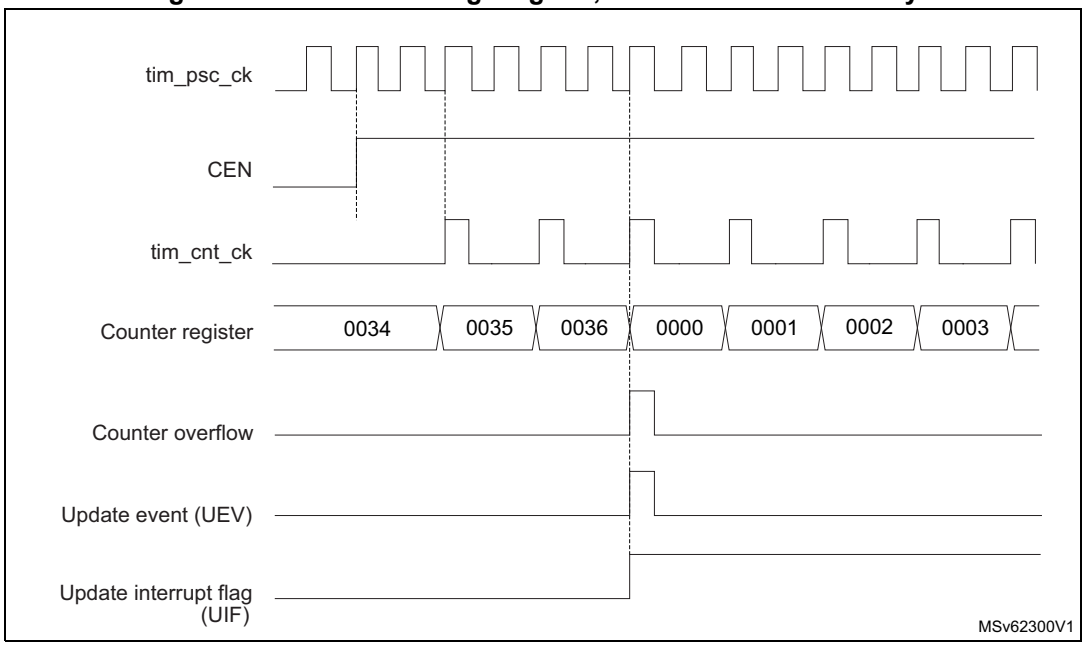


Figure 566. Counter timing diagram, internal clock divided by 4

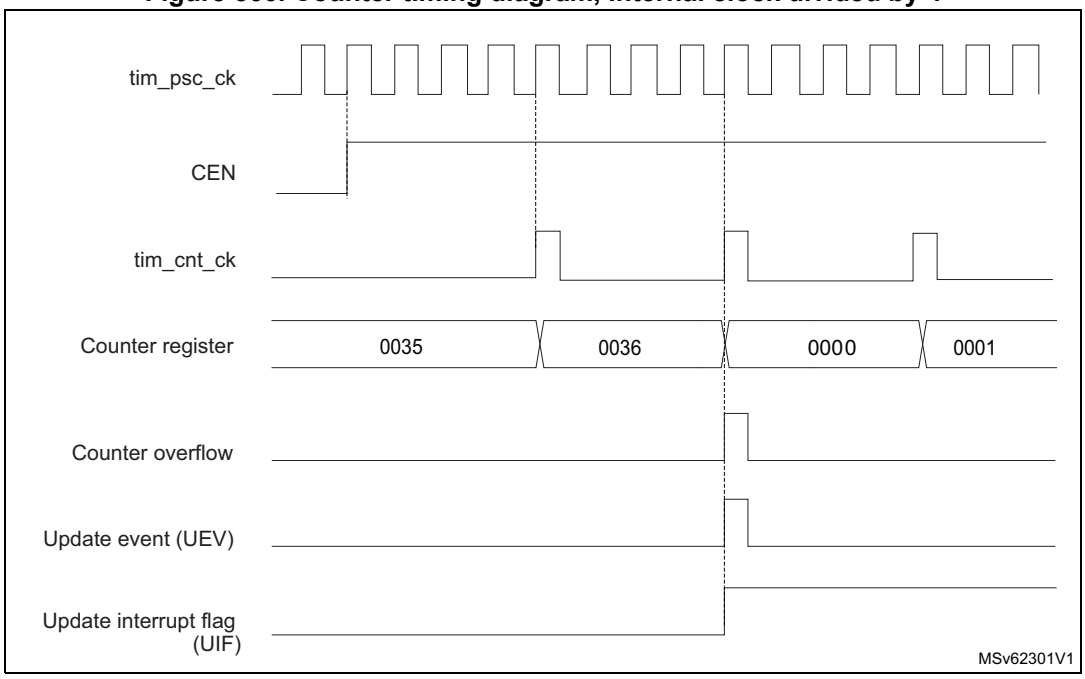


Figure 567. Counter timing diagram, internal clock divided by N

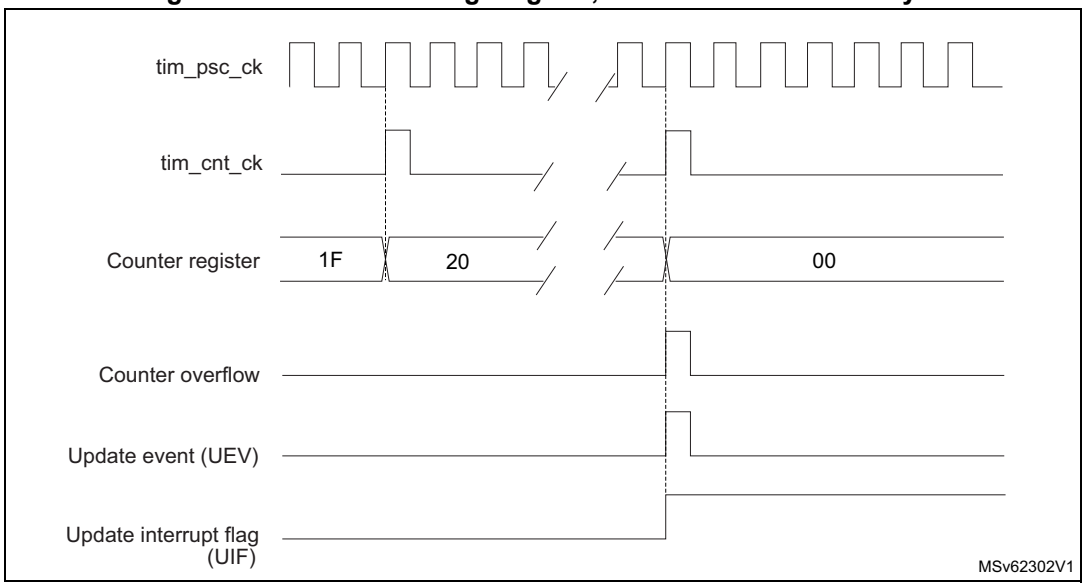


Figure 568. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

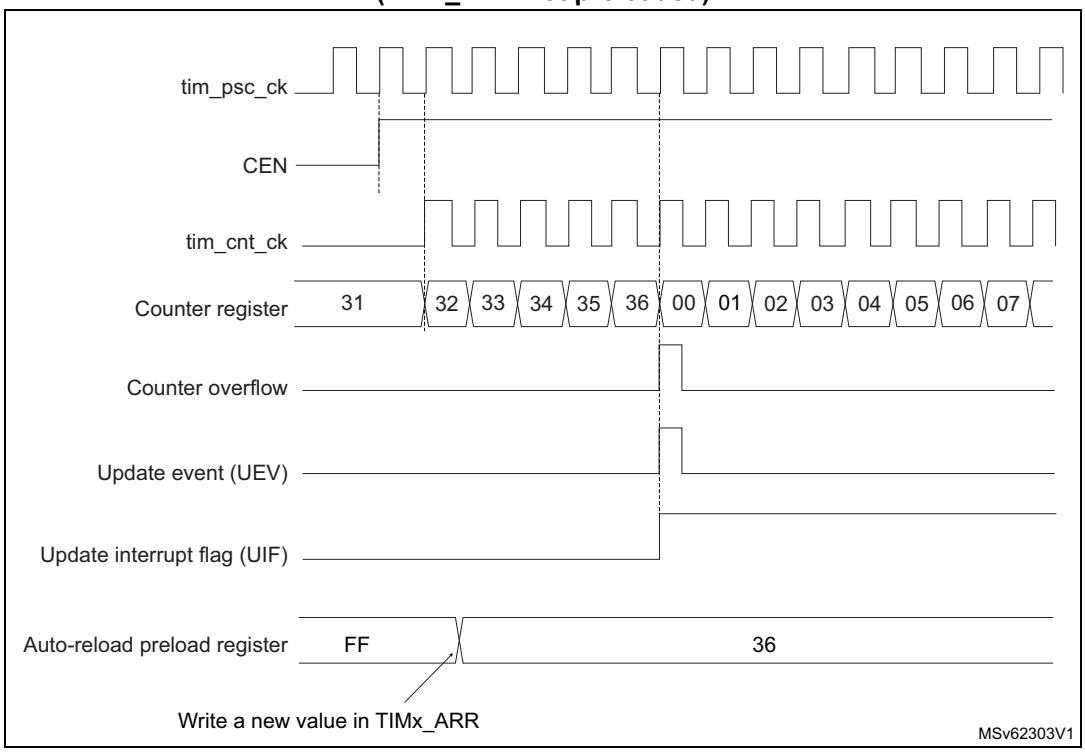
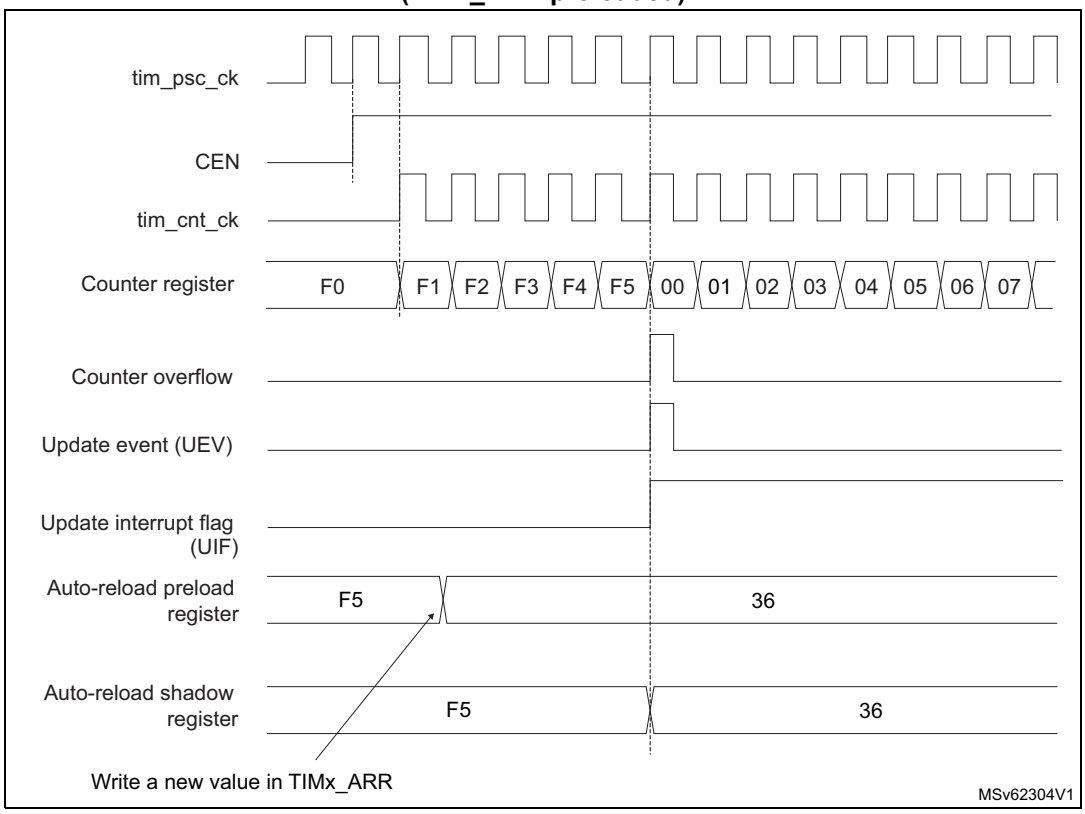


Figure 569. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



42.4.5 Repetition counter

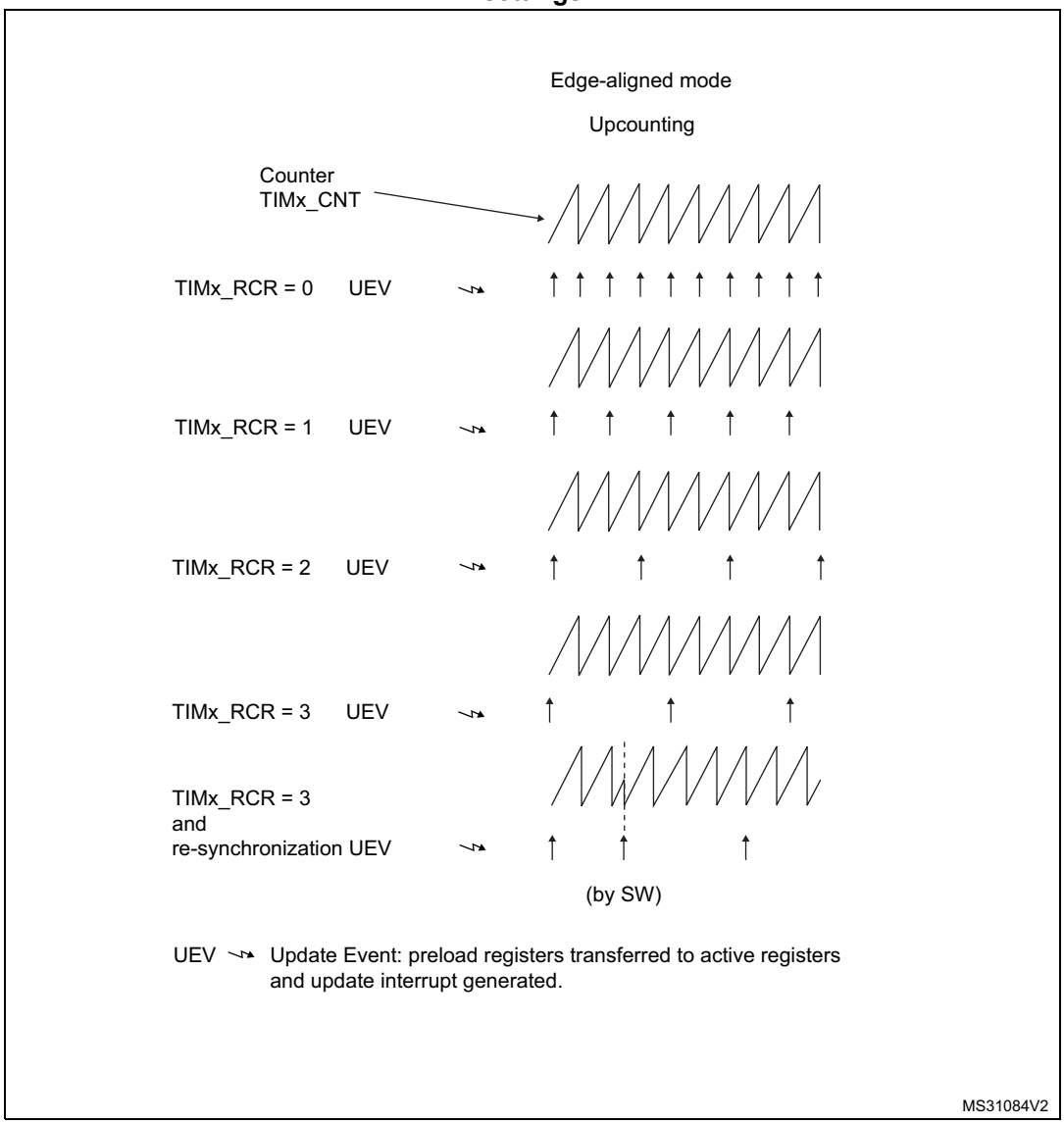
Section 42.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 570*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 570. Update rate examples depending on mode and TIMx_RCR register settings



42.4.6 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another timer* for more details.

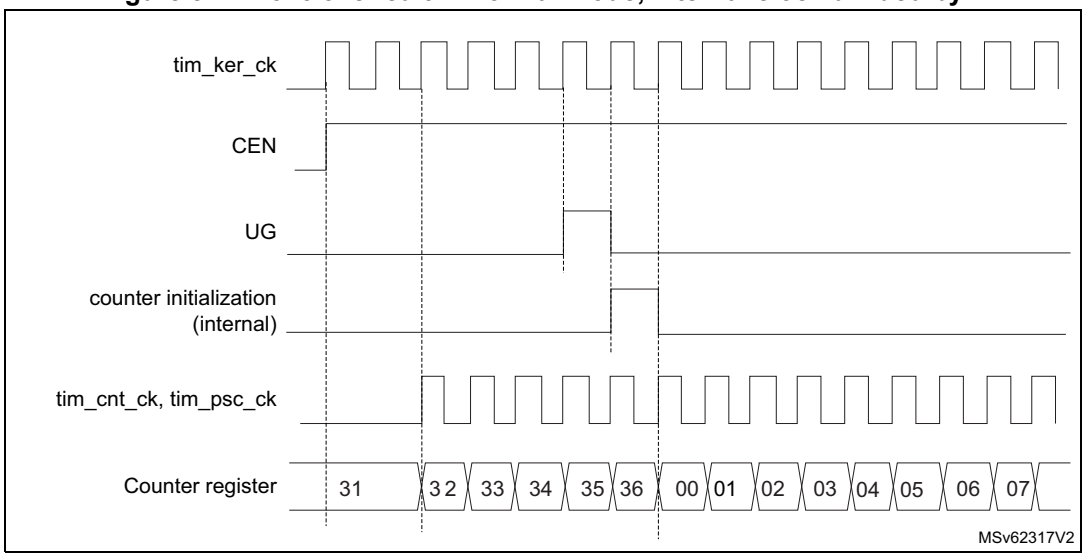
Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 571 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

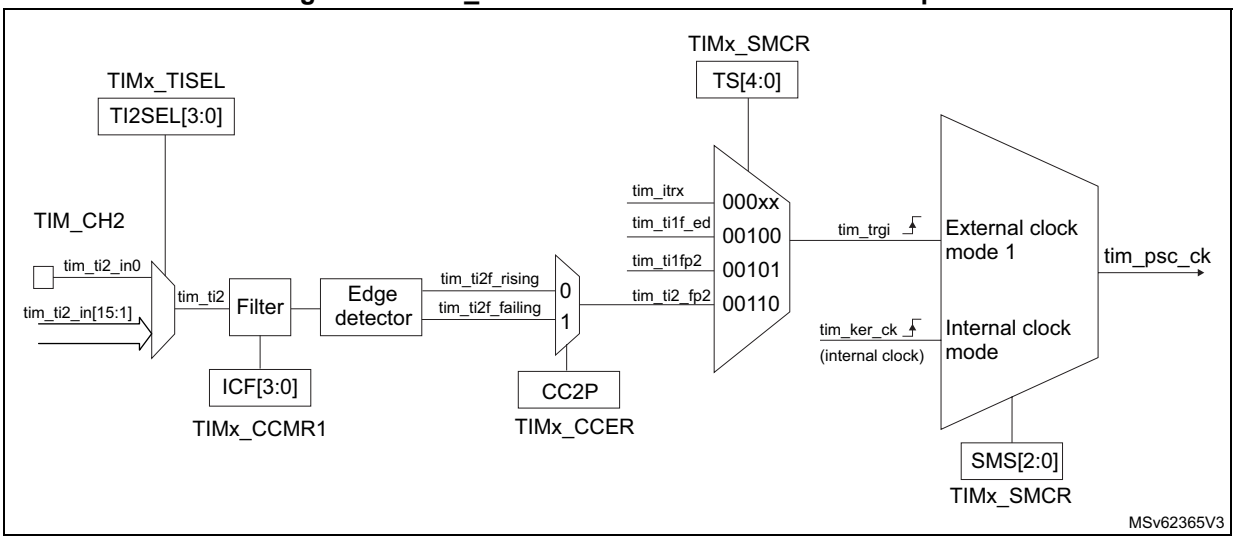
Figure 571. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 572. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

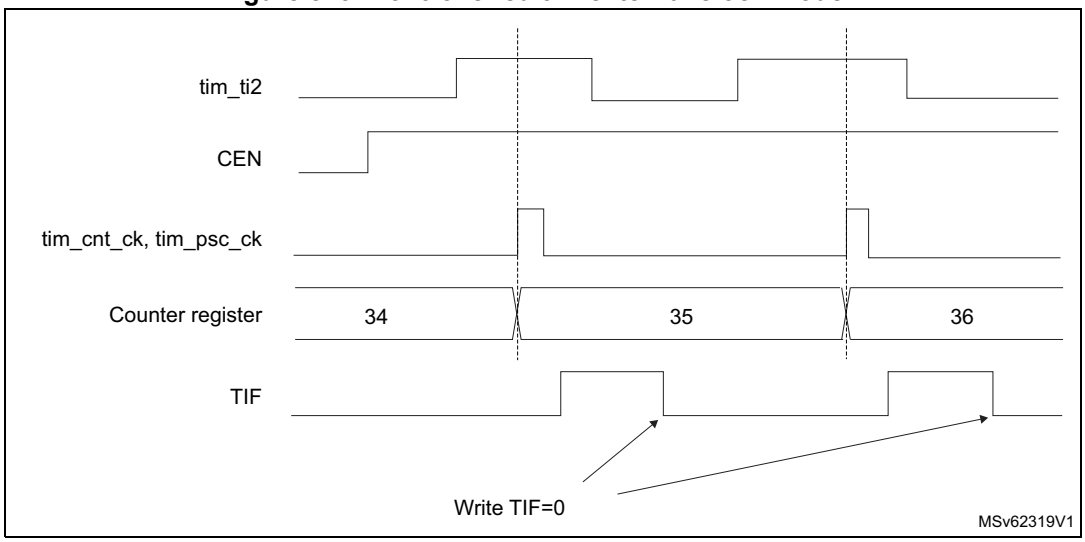
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 573. Control circuit in external clock mode 1



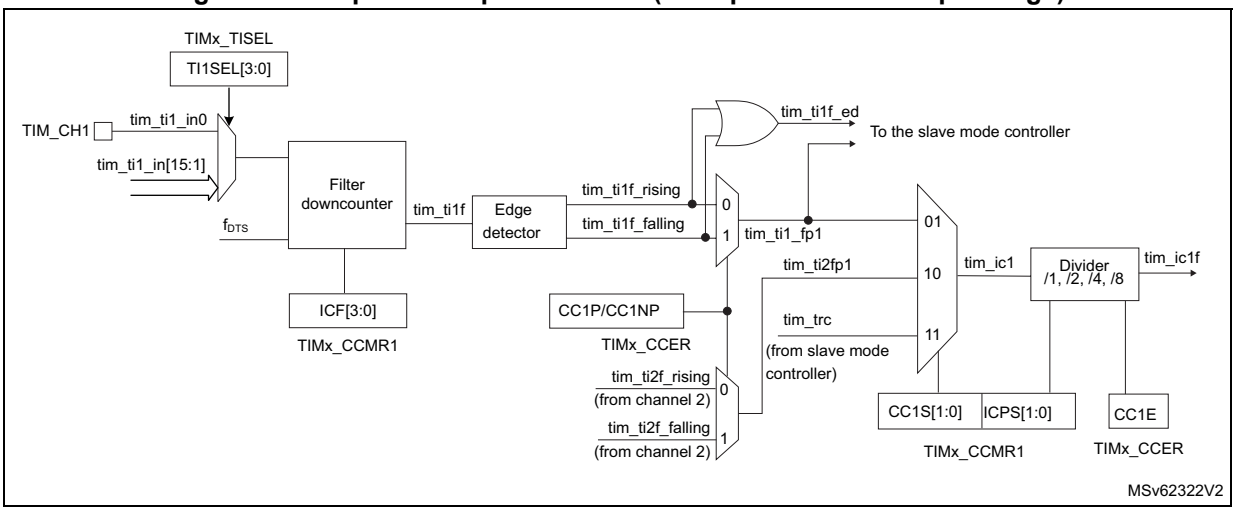
42.4.7 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 574 to *Figure 577* give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 574. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 575. Capture/compare channel 1 main circuit

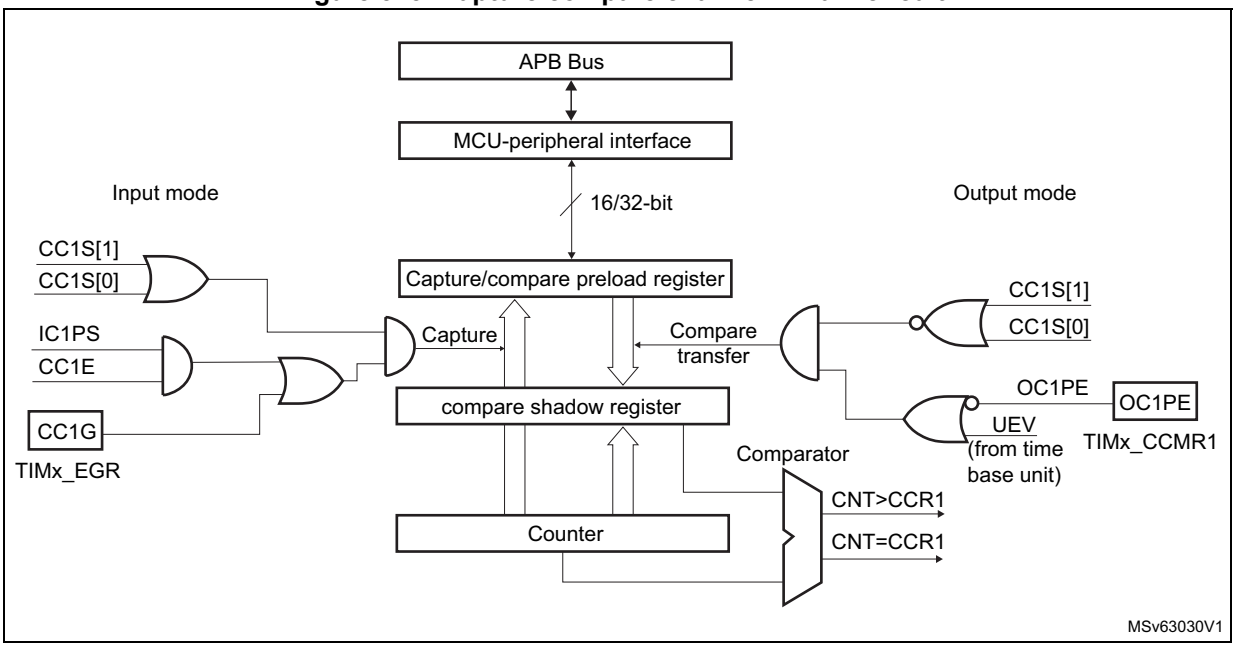


Figure 576. Output stage of capture/compare channel (channel 1)

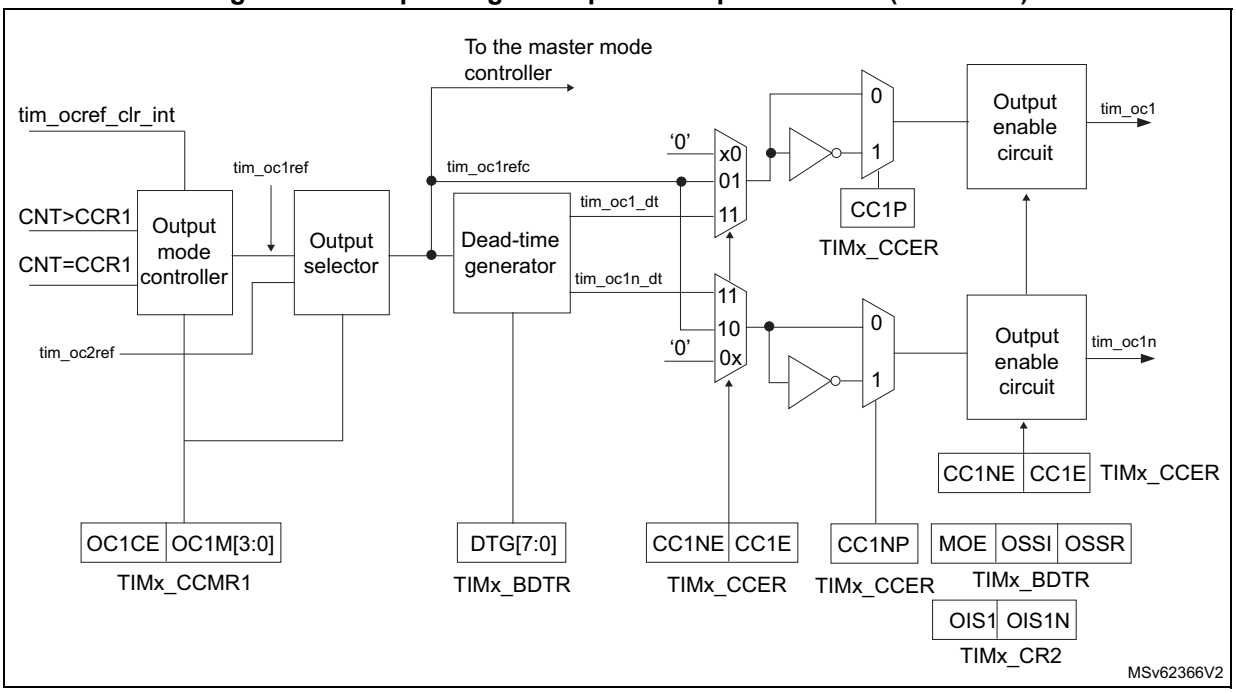
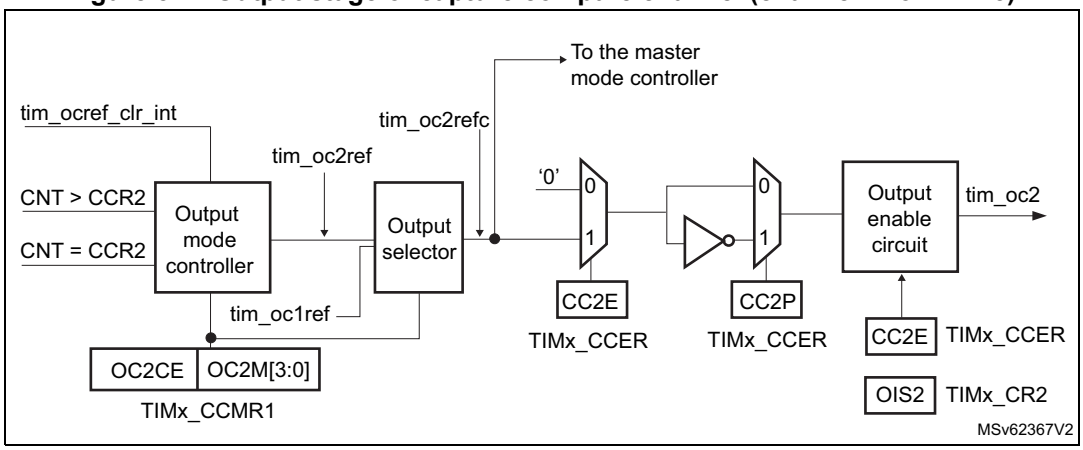


Figure 577. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

42.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was

already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

42.4.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

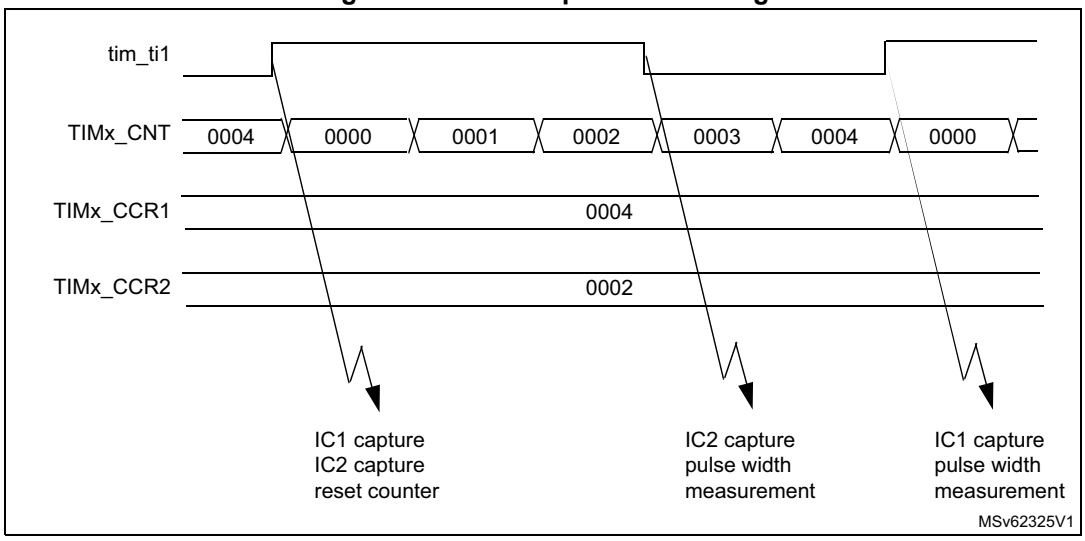
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 578. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

42.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

42.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

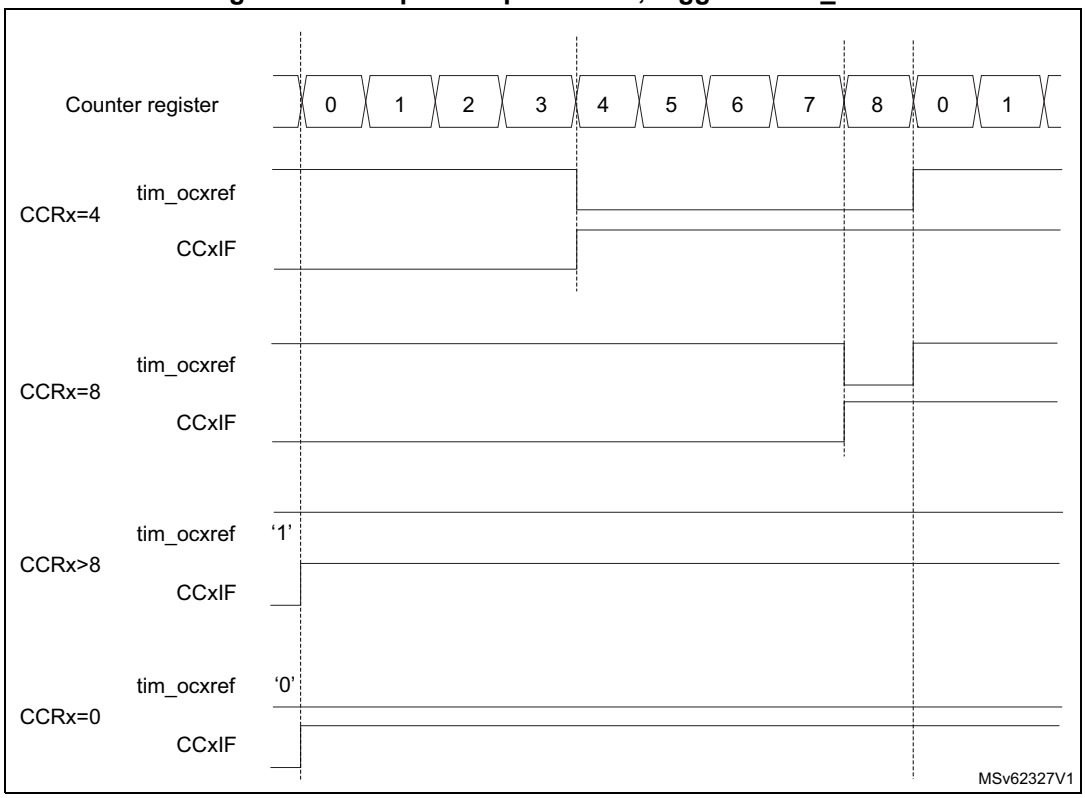
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 579*.

Figure 579. Output compare mode, toggle on tim_oc1



42.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

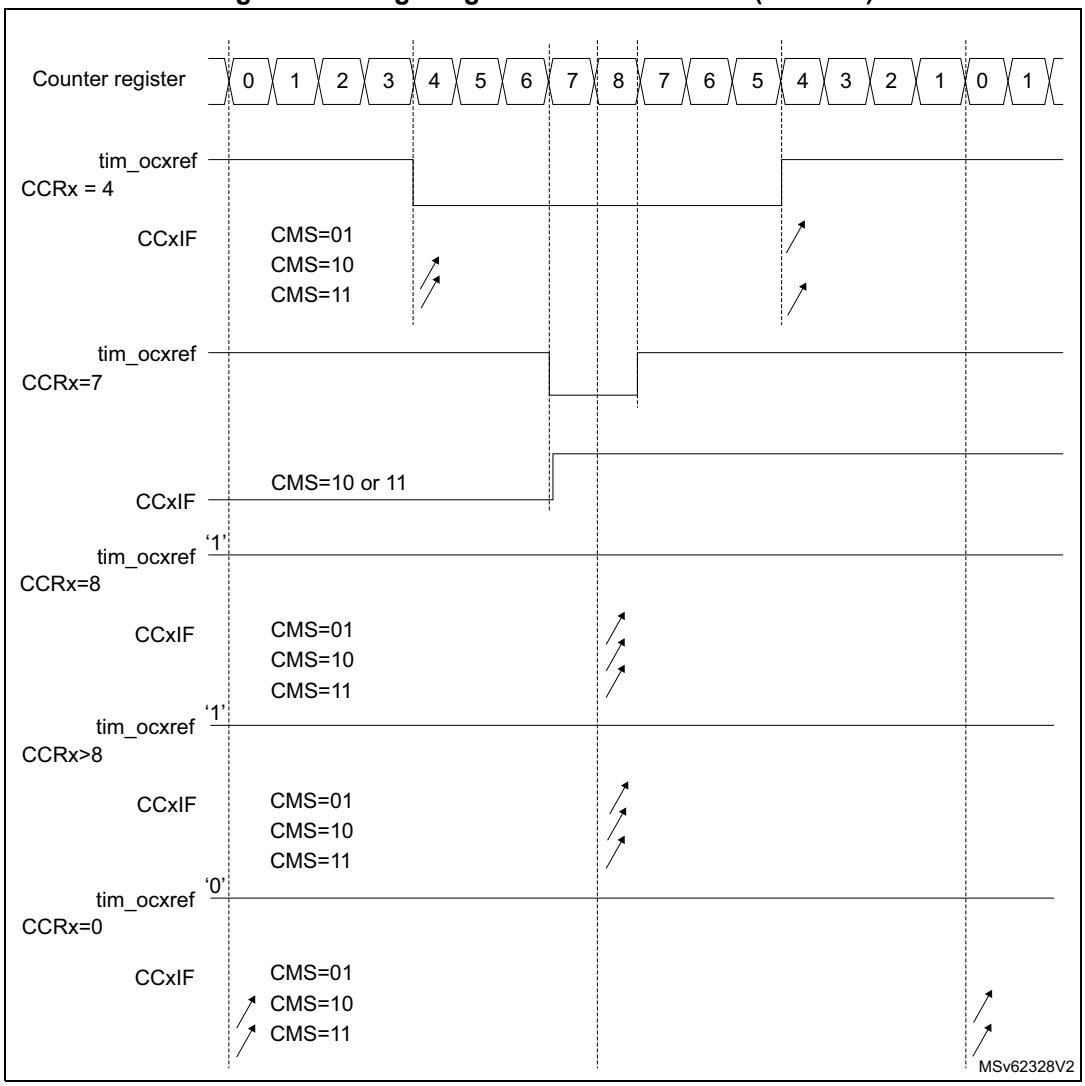
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode*.

In the following example applies to PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in

TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 580* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 580. Edge-aligned PWM waveforms (ARR = 8)

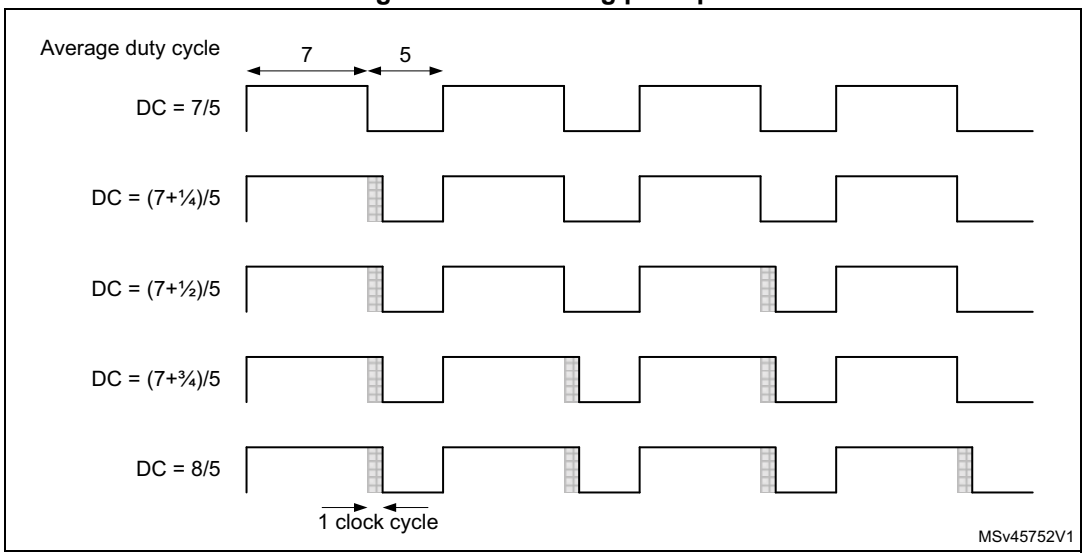


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 581* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 581. Dithering principle

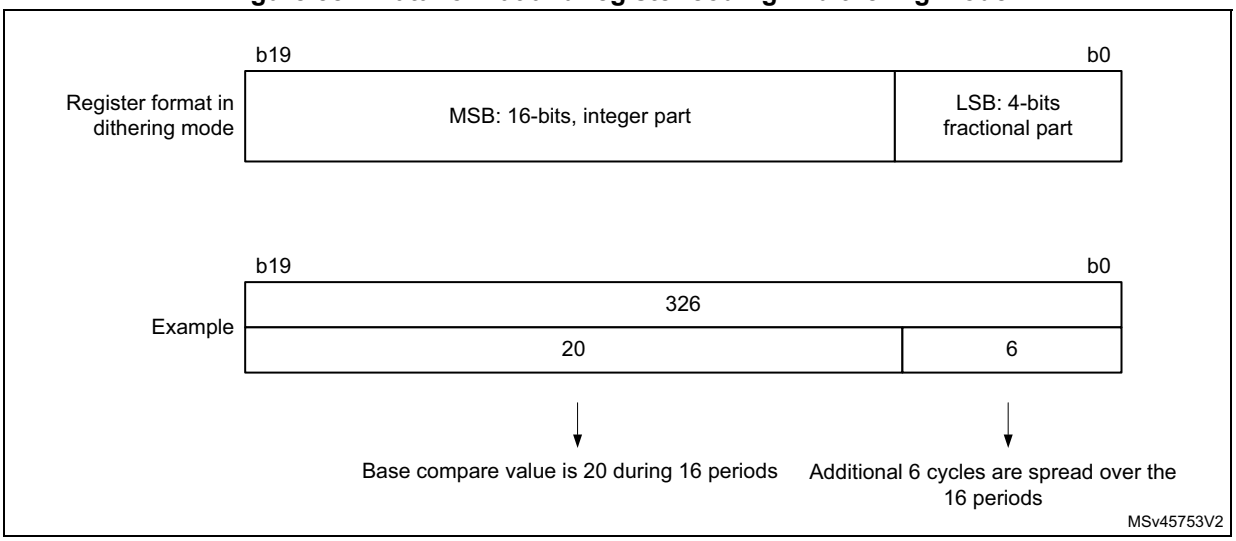


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 582* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 582. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

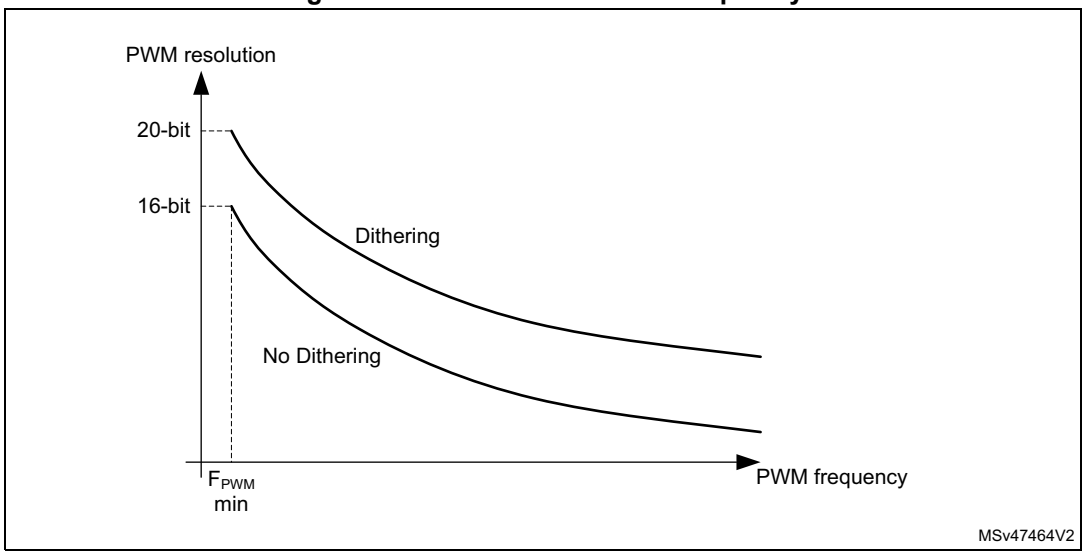
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

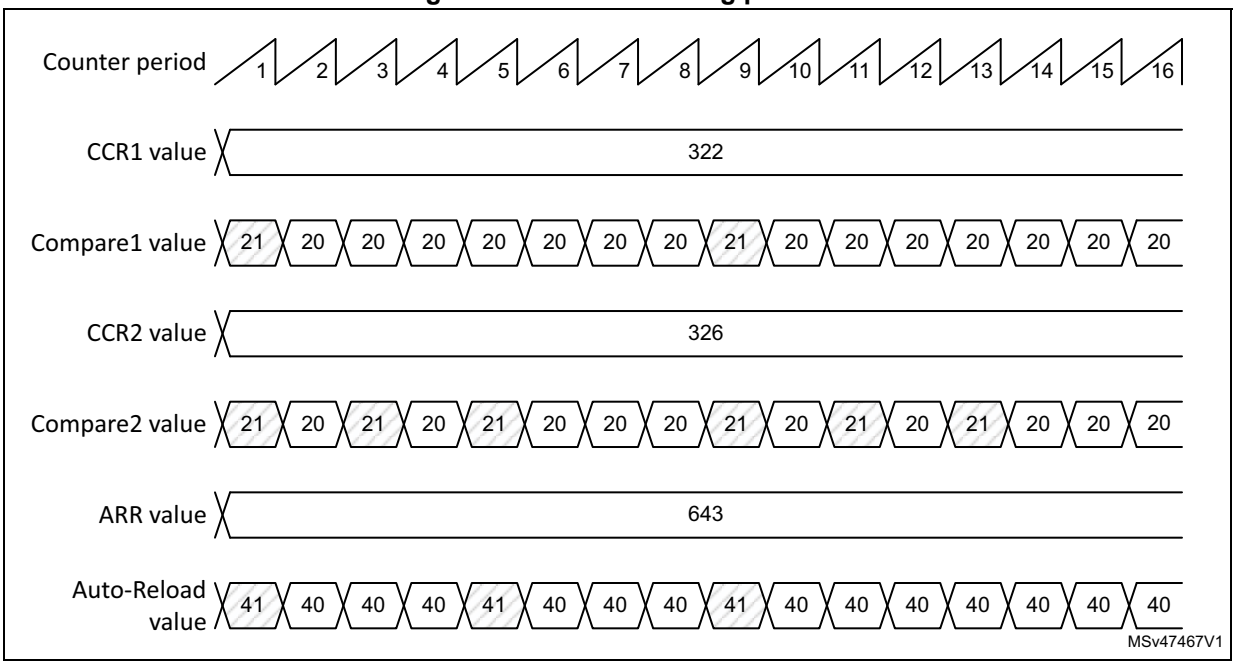
As shown on the *Figure 583* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 583. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 584* below.

Figure 584. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 457* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 457. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 457. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

42.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

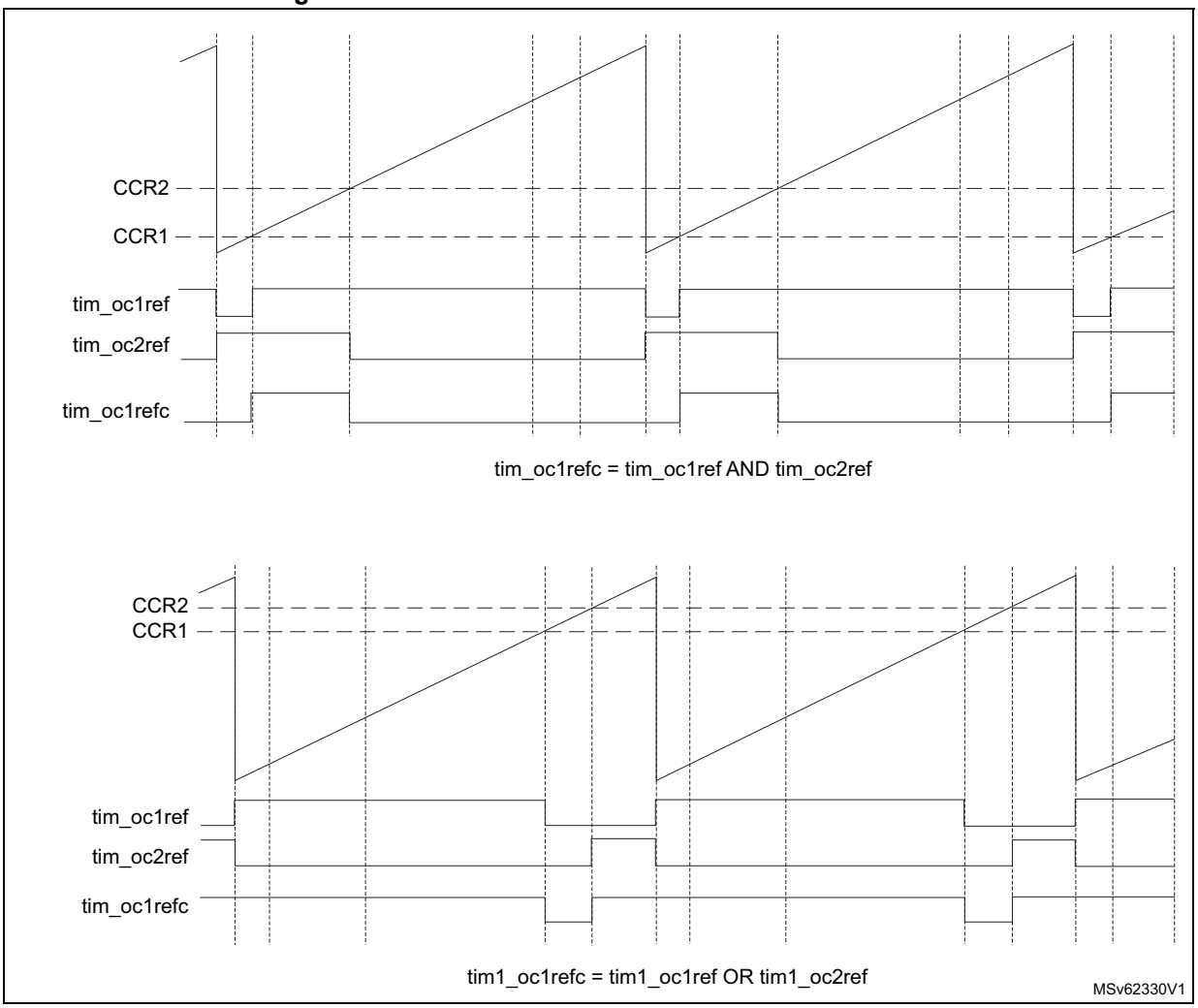
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Table 585 represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 585. Combined PWM mode on channel 1 and 2



42.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 464: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

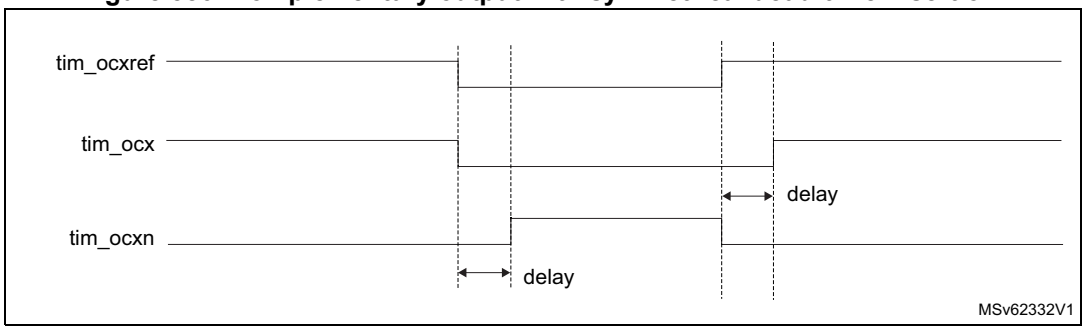
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 586. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 587*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 587. Asymmetrical deadtime

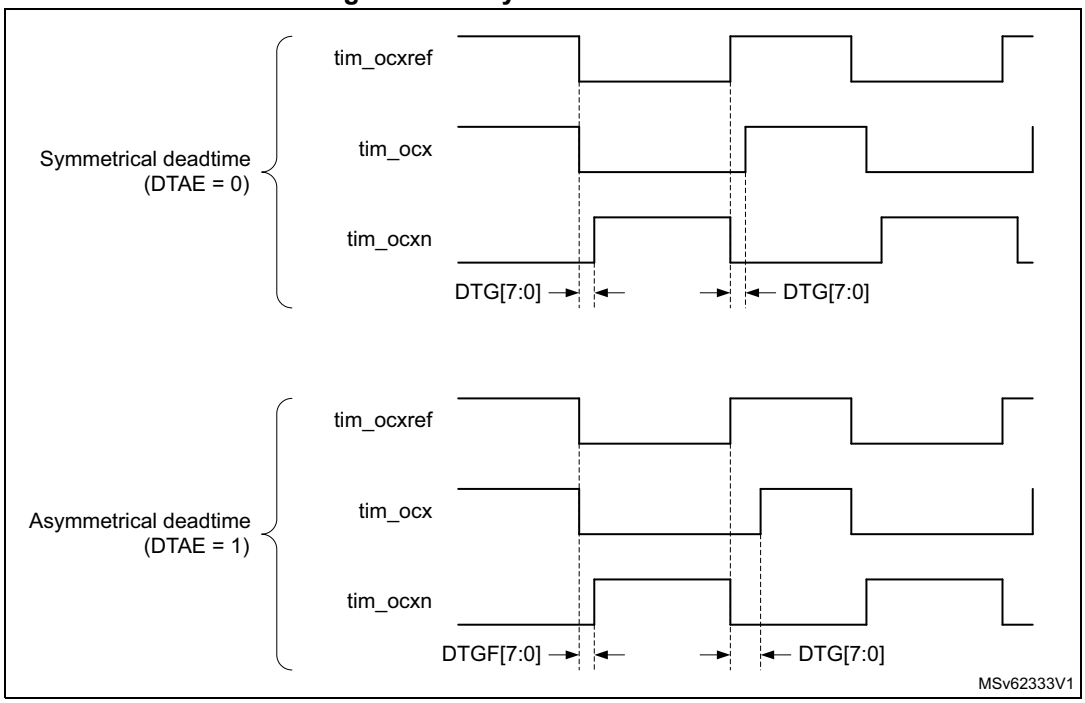


Figure 588. Dead-time waveforms with delay greater than the negative pulse.

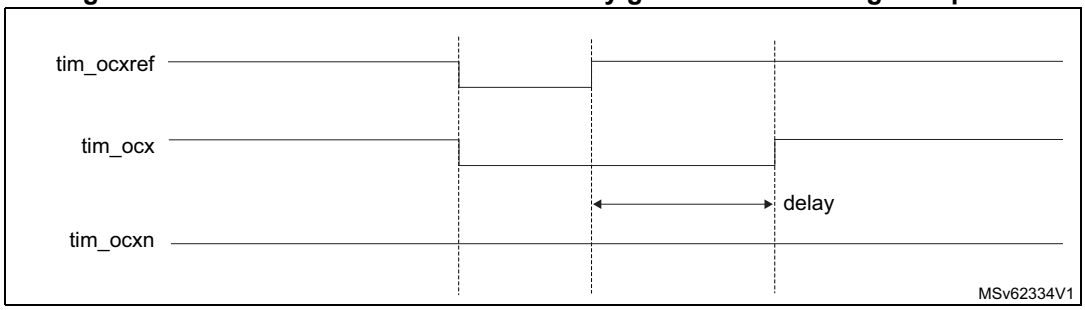
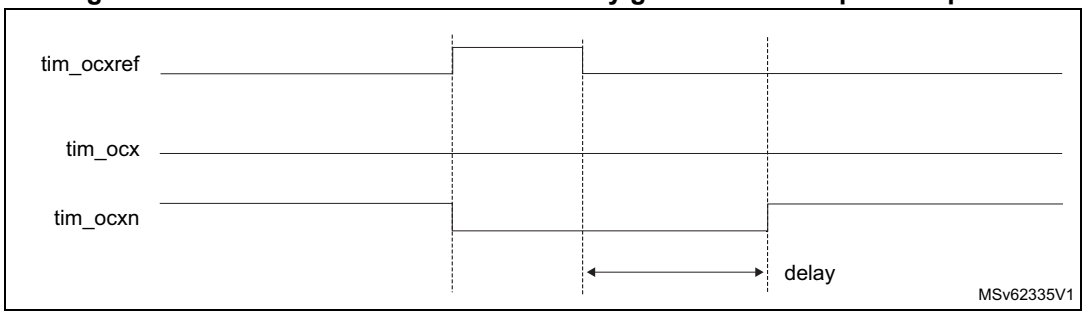


Figure 589. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 42.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

42.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 464: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIMx_BDTR register).
- Programmable enable bit (BKE bit in the TIMx_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

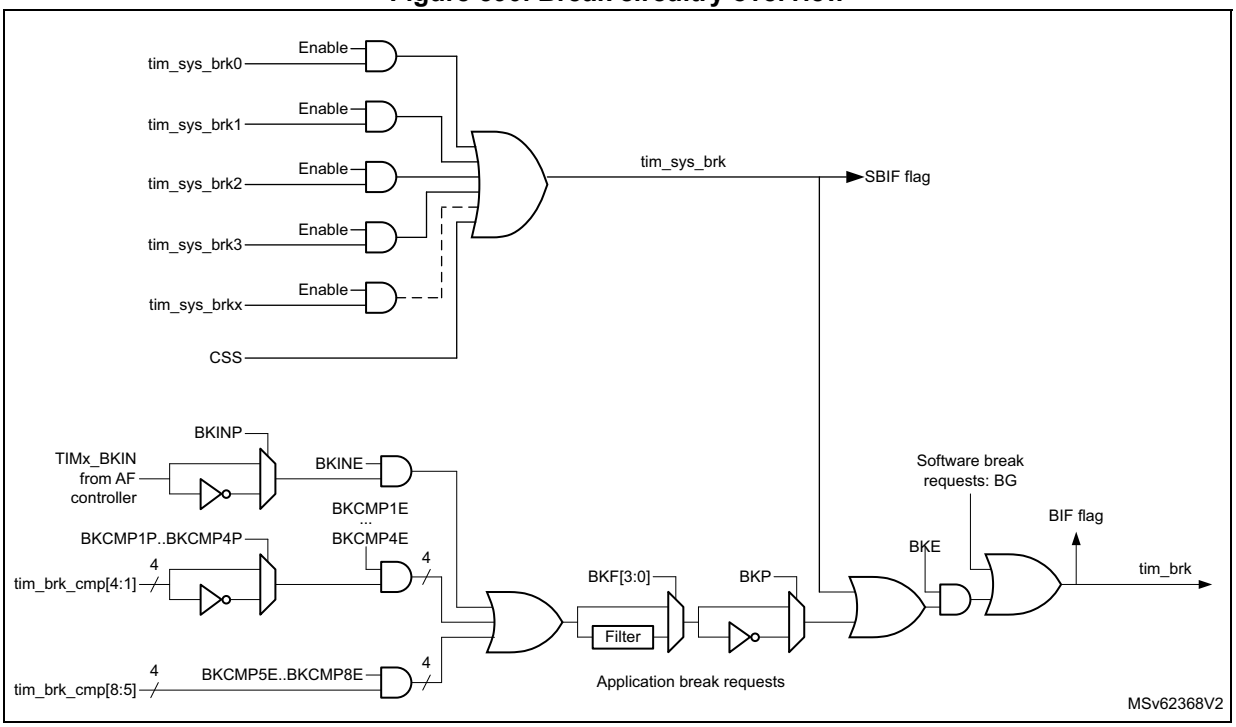
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 590* below.

Figure 590. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

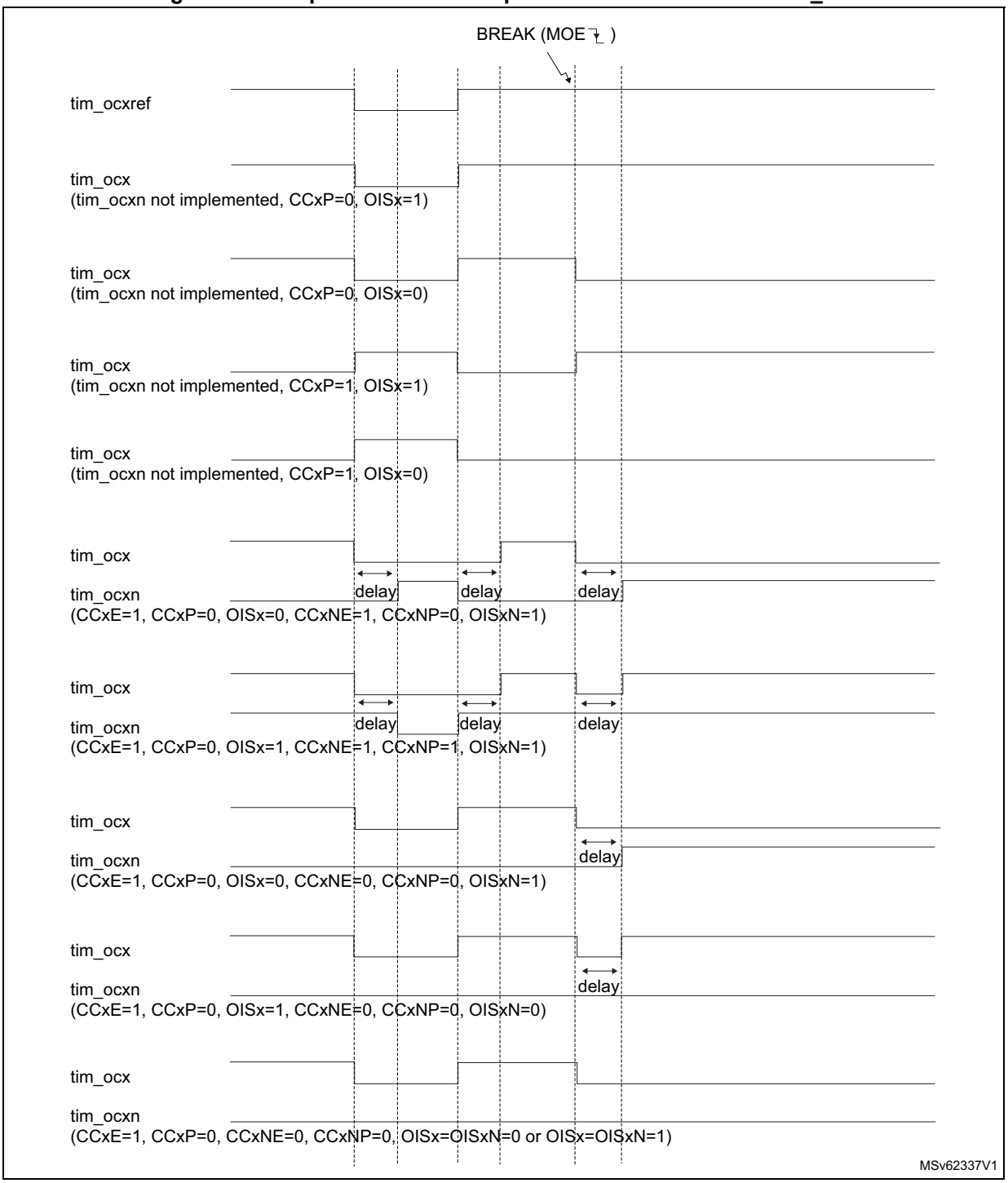
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 42.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 591* shows an example of behavior for the outputs in response to a break.

Figure 591. Output behavior in response to a break event on tim_brk



42.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 592*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 458*).

Table 458. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

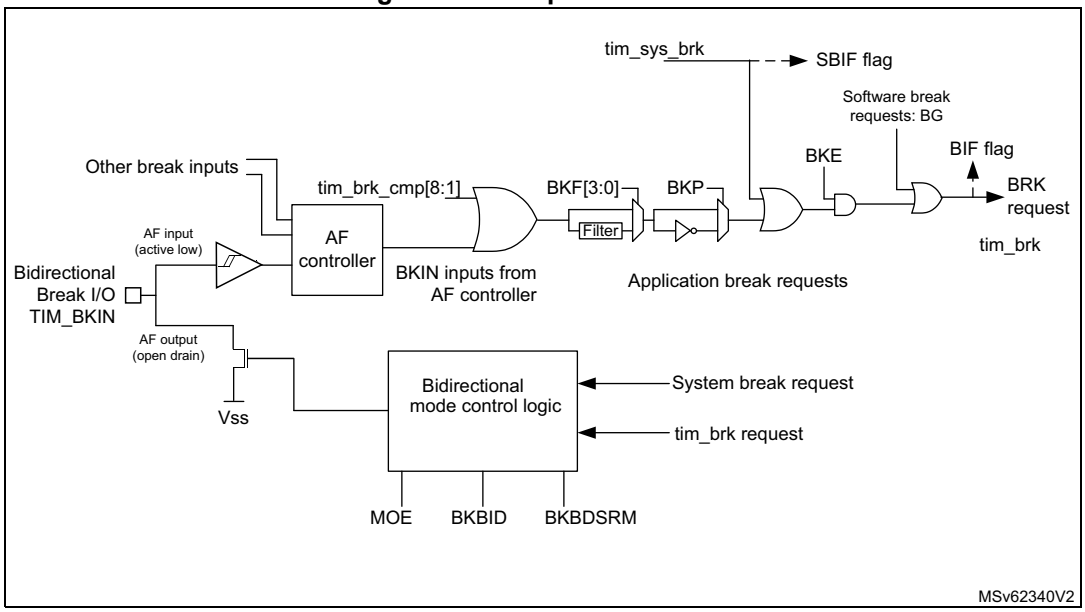
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 592. Output redirection

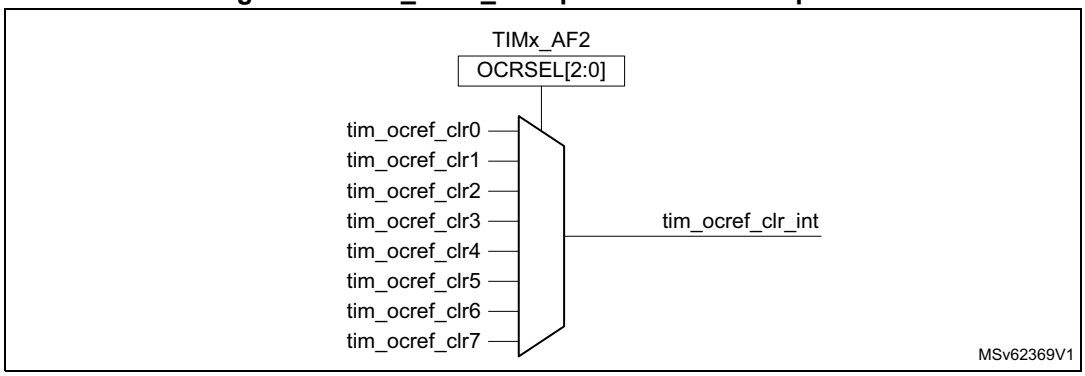


42.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 509* below.

Figure 593. tim_ocref_clr input selection multiplexer



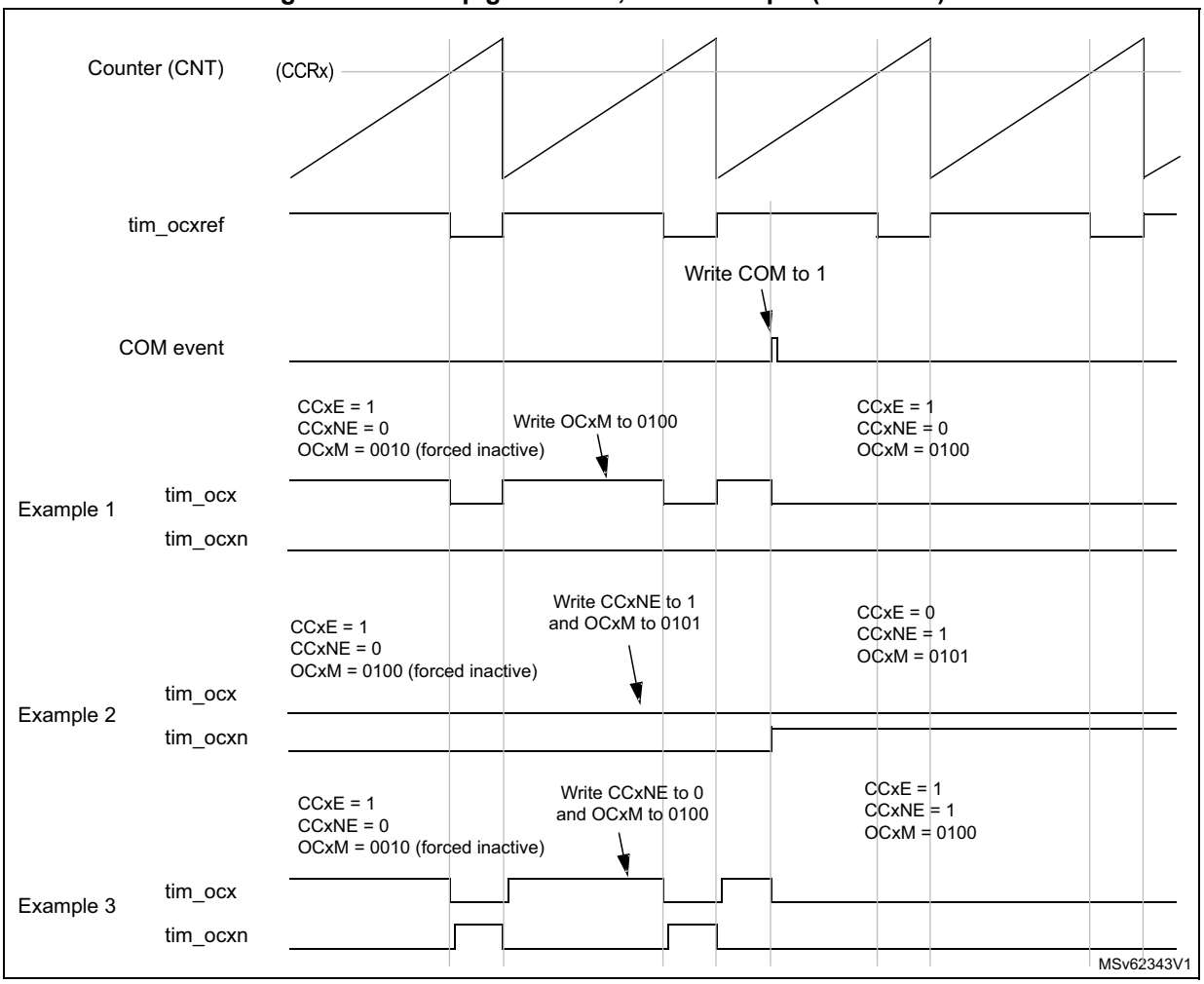
42.4.18 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 594* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 594. 6-step generation, COM example (OSSR = 1)



42.4.19 One-pulse mode

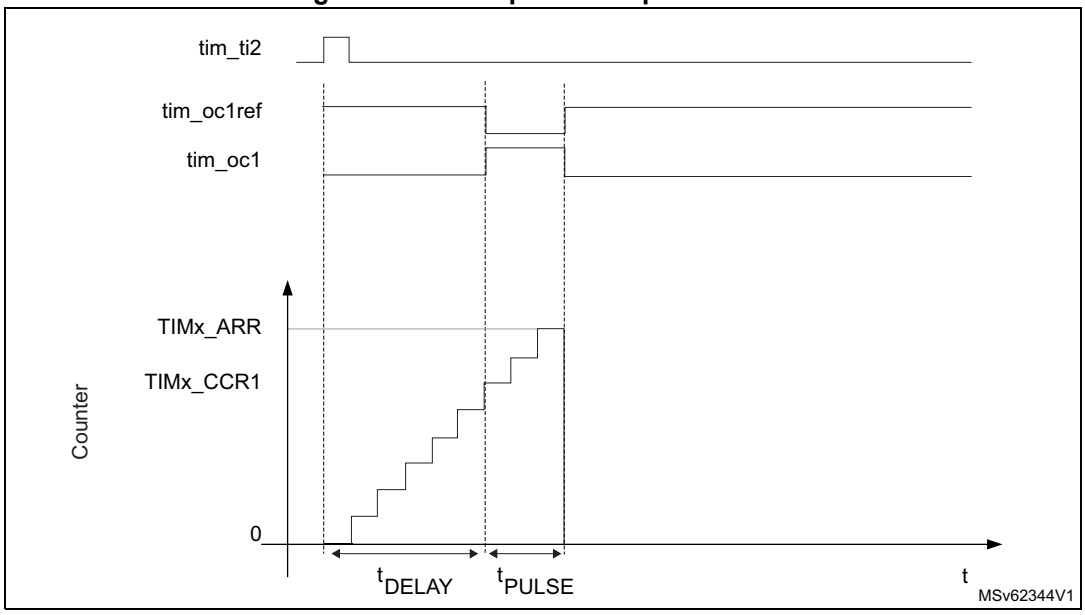
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx).

Figure 595. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

42.4.20 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 42.4.19*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

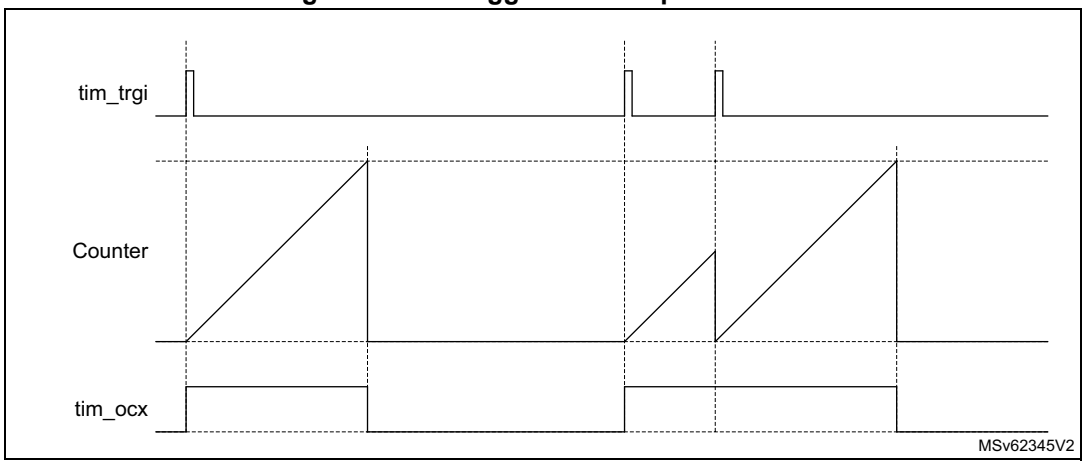
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 596. Retriggerable one pulse mode



42.4.21 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

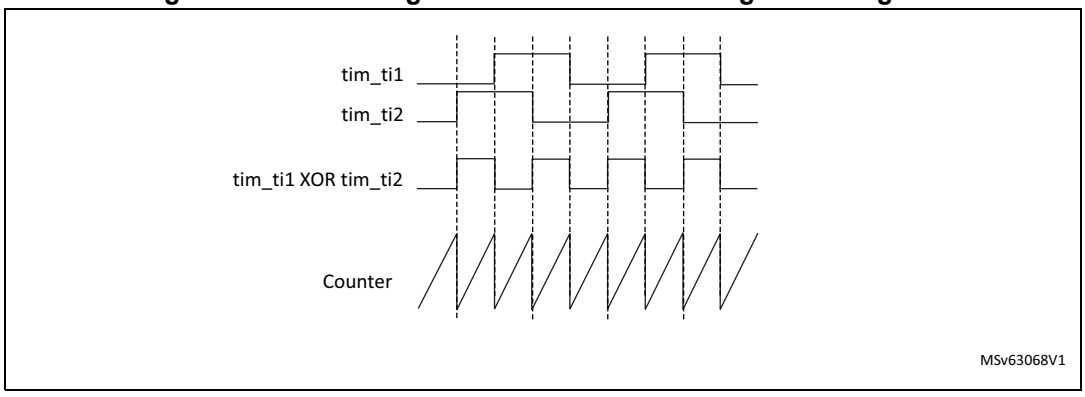
There is no latency between the assertions of the UIF and UIFCPY flags.

42.4.22 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 597*.

Figure 597. Measuring time interval between edges on 2 signals



42.4.23 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

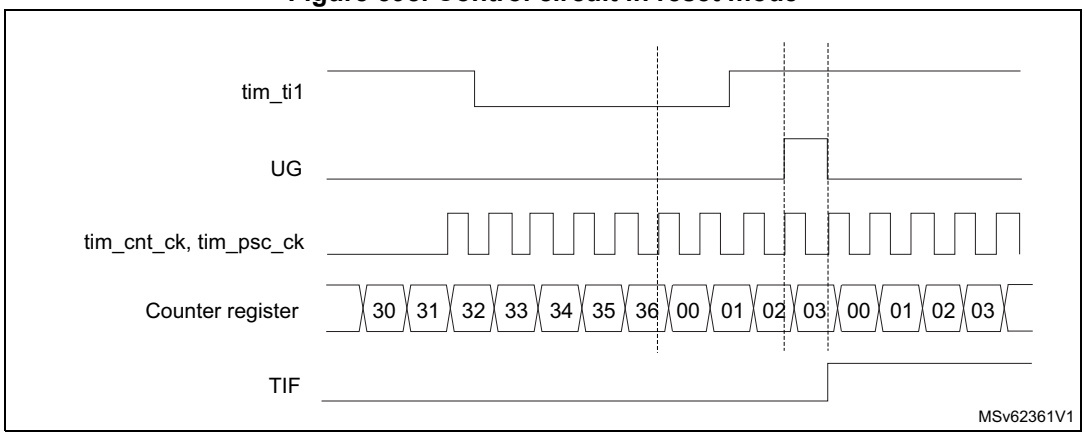
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 598. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

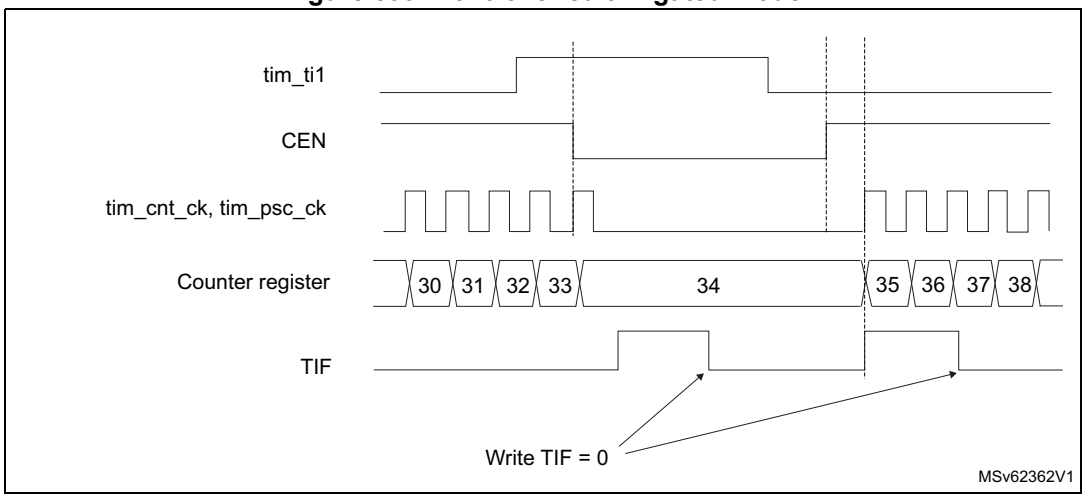
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 599. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

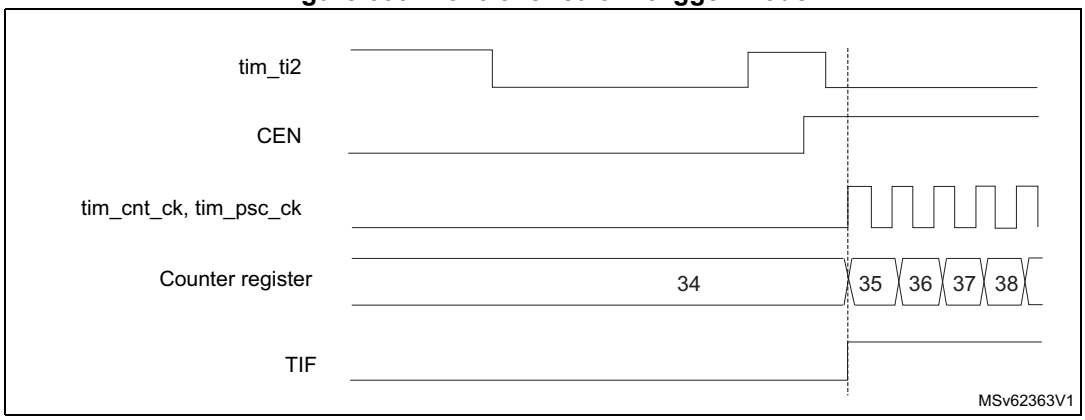
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 600. Control circuit in trigger mode



Slave mode selection preload for run-time update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

42.4.24 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

42.4.25 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

42.4.26 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 39.4.23: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

42.4.28 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.29 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 42.8.20: TIMx DMA control register (TIMx_DCR)(x = 16 to 17)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

42.4.30 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in *Table 459*.

Table 459. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_com_dma(1)	Commutation (COM)	COMDE
tim_trgi_dma(1)	Trigger	TDE

1. Available for TIM15 only.

42.4.31 Debug mode

When the microcontroller enters debug mode (Cortex-M33 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

42.5 TIM15/TIM16/TIM17 low-power modes

Table 460. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

42.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 461*.

Table 461. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

1. Available for TIM15 only.

42.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

42.7.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM15_ARR register is not buffered
1: TIM15_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt if enabled. These events can be:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

42.7.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)

0: tim_oc2 = 0 when MOE = 0

1: tim_oc2 = 1 when MOE = 0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

42.7.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
							rw			rw	rw				rw

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE:** SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded.

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw				rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

42.7.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected
1: The counter value has been captured in TIM15_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred

1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM15_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS = 0 and UDIS = 0 in the TIM15_CR1 register.

– When CNT is reinitialized by a trigger event (refer to *Section 42.7.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS = 0 and UDIS = 0 in the TIM15_CR1 register.

42.7.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation

Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1 A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

42.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM15_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

42.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM15_CNT = TIM15_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

42.7.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 462* for details.

Table 462. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0 tim_ocxn = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP Then if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

42.7.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM15_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

42.7.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in “reset mode”).

42.7.12 TIM15 autoreload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 42.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

42.7.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

42.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

42.7.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

42.7.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM15_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 42.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1845*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1; Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 42.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1845*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 42.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 1845*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. tim_ocx/tim_ocxn enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM15_BDTR register, OISx and OISxN bits in TIM15_CR2 register and BKBID/BKE/BKP/AOE bits in TIM15_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM15_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM15_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM15_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS} = 125 ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

42.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

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

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable
0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register
1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup
This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.
DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.
DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.
DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.
DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.
Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps
Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

42.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input
0000: TIM15_CH2 input (tim_ti2_in0)
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIM15_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

42.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

42.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

42.7.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

0110: COM

0111: Trigger

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM15_DMAR address).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:

00000: TIM15_CR1,
00001: TIM15_CR2,
00010: TIM15_SMCR,
...

42.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM15_CR1 address) + (DBA + DMA index) x 4

where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

42.7.23 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 463. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**TIM15_CR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	**TIM15_CR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value																						0	0	0	0	0	0	0	0	0		0
0x08	**TIM15_SMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value								0			0	0				0								0	0	0	0			0	0	0	0
0x0C	**TIM15_DIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	TIE	COMIE	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0					0	0	0	0	0		0	0	0
0x10	**TIM15_SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0
0x14	**TIM15_EGR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0

Table 463. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	**TIM15_CCMR1** Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM15_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	**TIM15_CCER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	**TIM15_CNT**	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	**TIM15_PSC**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	**TIM15_ARR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**TIM15_RCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**TIM15_CCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**TIM15_CCR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	**TIM15_BDTR**	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	**TIM15_DTR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															

Table 463. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C	**TIM15_TISEL**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	**TIM15_AF1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	**TIM15_AF2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68 - 0x3D8	Reserved	Res.																															
0x3DC	**TIM15_DCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	**TIM15_DMAR**	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

42.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

42.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

42.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

42.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

42.8.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

42.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

42.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

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

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

42.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

42.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 464* for details.

Table 464. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CC1E bit	CC1NE bit	tim_oc1 output state	tim_oc1n output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0 tim_oc1n = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_oc1 = 0	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		0	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Output Disabled (not driven by the timer: Hi-Z) tim_oc1n = 0
		X	1	1	tim_oc1ref + Polarity + dead-time	Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_oc1 = CC1P	tim_oc1ref + Polarity tim_oc1n = tim_oc1ref XOR CC1NP
		1	1	0	tim_oc1ref + Polarity tim_oc1 = tim_oc1ref XOR CC1P	Off-State (output enabled with inactive state) tim_oc1n = CC1NP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

42.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

42.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

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

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

42.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0x0000 FFFF

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 42.4.3: Time-base unit on page 1790* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

42.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

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

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

42.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

42.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 42.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 42.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 42.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

42.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054

Reset value: 0x0000 0000

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

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

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

42.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

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

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

42.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060

Reset value: 0x0000 0001

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

42.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

42.8.19 TIM17 option register 1 (TIM17_OR1)

Address offset: 0x68

Reset value: 0x0000 0000

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

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

Bit 1:0 **RTCPREEN[1:0]**: RTCPRE divider enable

This bit enables the RTCPRE HSE divider for the tim_ti1_in2. See *Section 42.4.2: TIM15/TIM16/TIM17 pins and internal signals* and *Section 15: Peripherals interconnect matrix*.

10: RTCPRE HSE divider for the tim_ti1_in2 enabled

Others: RTCPRE HSE divider for the tim_ti1_in2 disabled

42.8.20 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x3DC

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

42.8.21 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0

Reset value: 0x0000 0000

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

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

42.8.22 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 465. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE		COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]				OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																0									0	0	0	0		0	0	0	0
0x1C	Reserved	Res.																															
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 465. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIMx_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0x64	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68	TIMx_OR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCPREEN[1:0]	
	Reset value																															0	0
0x6C - 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

43 Low-power timer (LPTIM)

43.1 LPTIM introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

43.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

43.3 LPTIM implementation

Table 466 describes LPTIM implementation on STM32H5 devices. The full set of features is implemented in LPTIM1, 2, 3, 5 and 6. LPTIM4 supports a smaller set of features.

Table 466. STM32H5 LPTIM features(1)

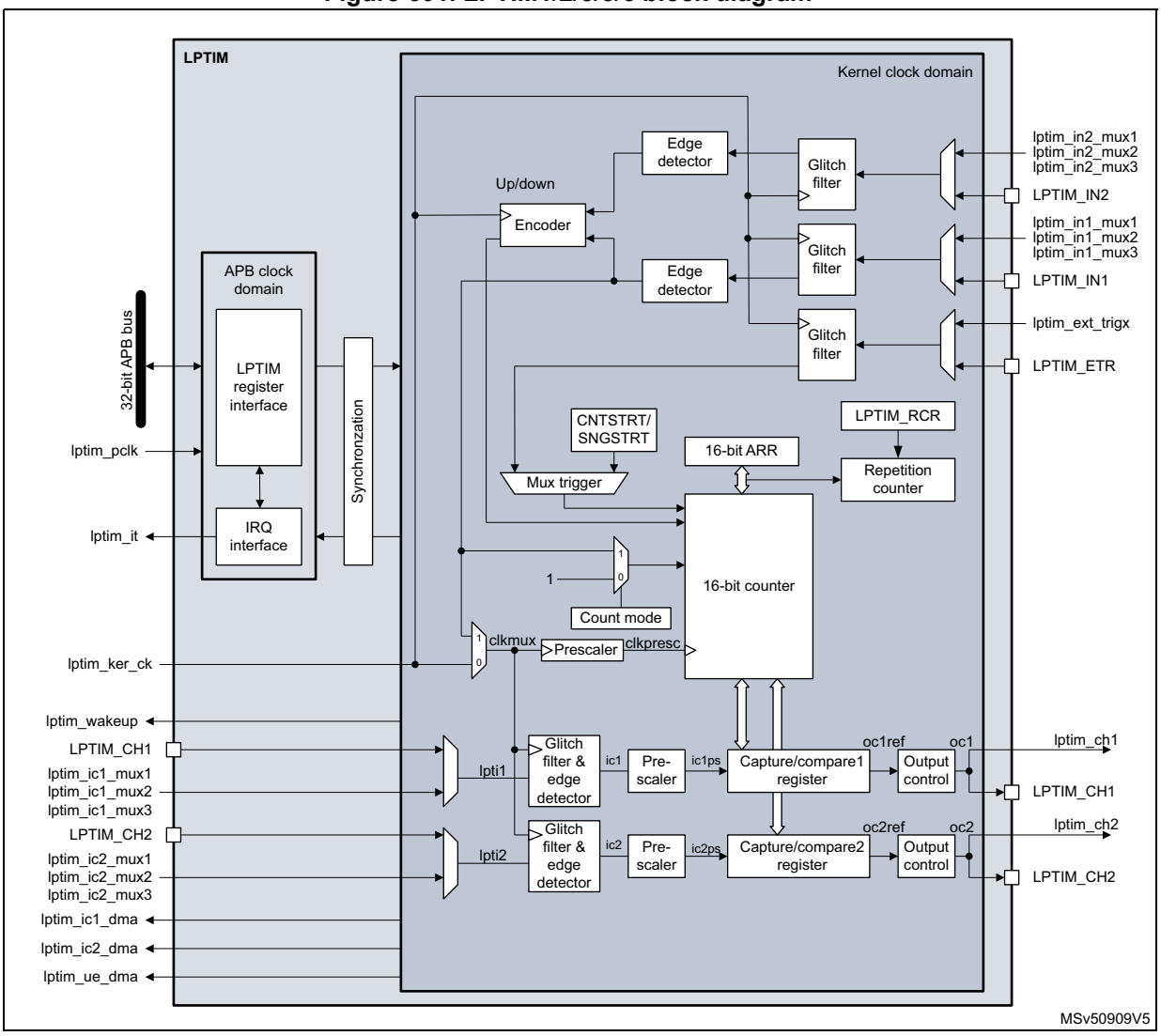
LPTIM modes/features(2)	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5	LPTIM6
Encoder mode	X	X	X	-	X	X
PWM mode	X	X	X	X	X	X
Input Capture	X	X	X	-	X	X
Number of channels	2	2	2	0	2	2
Number of DMA requests	3	3	3	0	3	3
Wake-up from Stop mode	X	X	X	X	X	X
Autonomous mode	-	-	-	-	-	-

1. STM32H523/533 devices implement only LPTIM1/2.
2. X = supported.

43.4 LPTIM functional description

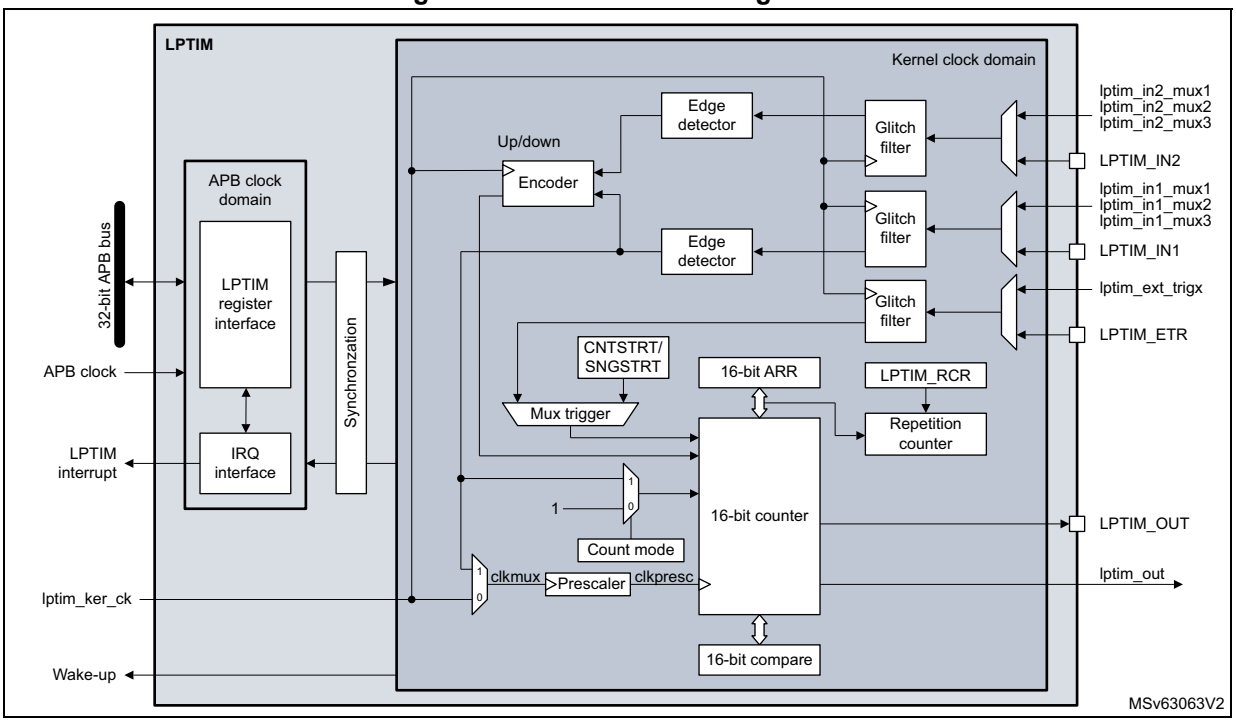
43.4.1 LPTIM block diagram

Figure 601. LPTIM1/2/3/5/6 block diagram(1)



1. Some I/Os may not be available, refer to *Section 43.4.2: LPTIM pins and internal signals*.

Figure 602. LPTIM4 block diagram(1)



1. Some I/Os may not be available, refer to *Section 43.4.2: LPTIM pins and internal signals*.

43.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 467. LPTIM1/2/3/5/6 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

Table 468. LPTIM4 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM Output GPIO pin

Table 469. LPTIM1/2/3/5/6 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

Table 470. LPTIM4 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event

43.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 471. LPTIM1/2/3/4/5/6 external trigger connections

TRIGSEL	External trigger					
	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5	LPTIM6
lptim_ext_trig0	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
lptim_ext_trig1	rtc_alra_trg	rtc_alra_trg	rtc_alra_trg	rtc_alra_trg	rtc_alra_trg	rtc_alra_trg
lptim_ext_trig2	rtc_alrb_trg	rtc_alrb_trg	rtc_alrb_trg	rtc_alrb_trg	rtc_alrb_trg	rtc_alrb_trg
lptim_ext_trig3	tamp_trg1	tamp_trg1	tamp_trg1	tamp_trg1	tamp_trg1	tamp_trg1
lptim_ext_trig4	tamp_trg2	gpdma_ch0 _tcf	Reserved	Reserved	Reserved	Reserved
lptim_ext_trig5	gpdma_ch1 _tcf	gpdma_ch4 _tcf	Reserved	Reserved	Reserved	Reserved
lptim_ext_trig6	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
lptim_ext_trig7	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Table 472. LPTIM1/2/3/5/6 input 1 connections

lptim_in1_mux	LPTIM1/2/3/4 input 1 connected to
lptim_in1_mux0	GPIO
lptim_in1_mux1	Reserved
lptim_in1_mux2	Reserved
lptim_in1_mux3	Reserved

Table 473. LPTIM1/2/3/5/6 input 2 connections

lptim_in2_mux	LPTIM1/2 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	Reserved
lptim_in2_mux2	Reserved
lptim_in2_mux3	Reserved

Table 474. LPTIM1/2/3/5/6 input capture 1 connections

lptim_ic1_mux	LPTIM1/2/3 input capture 1 connected to
lptim_ic1_mux0	GPIO
lptim_ic1_mux1	Reserved
lptim_ic1_mux2	Reserved
lptim_ic1_mux3	Reserved

Table 475. LPTIM1 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	LSI
lptim_ic2_mux2	LSE
lptim_ic2_mux3	Reserved

Table 476. LPTIM2 input capture 2 connections

lptim_ic2_mux	LPTIM2 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	HSI/1024
lptim_ic2_mux2	CSI/128
lptim_ic2_mux3	HSI/8

Table 477. LPTIM3/5/6 input capture 2 connections

lptim_ic2_mux	LPTIM3 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	Reserved
lptim_ic2_mux2	Reserved
lptim_ic2_mux3	Reserved

43.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

43.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

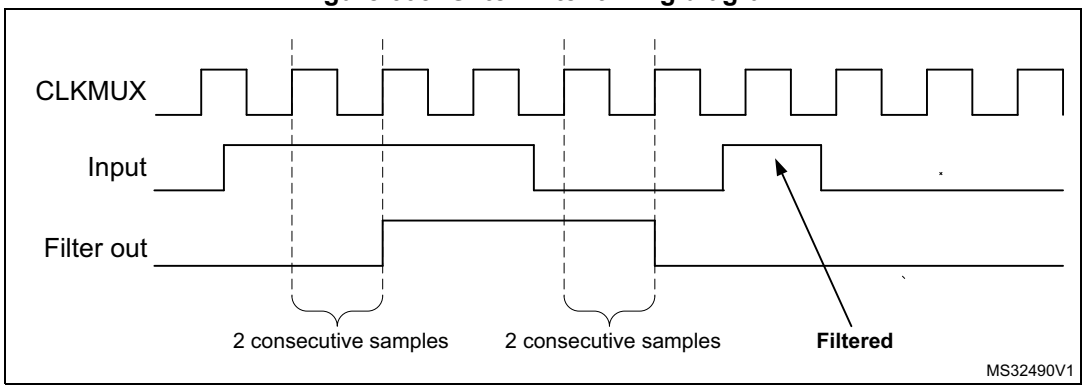
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 603* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 603. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

43.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 478. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

43.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

43.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

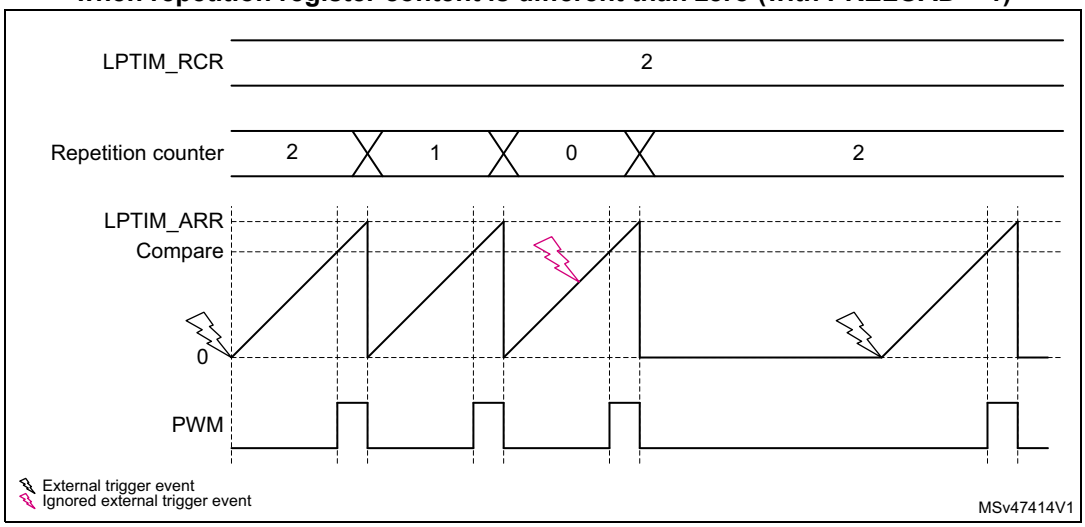
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 604*.

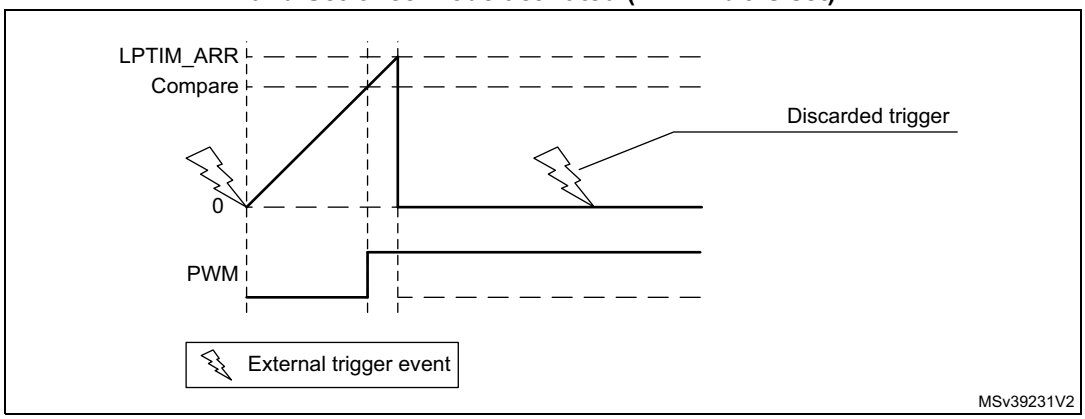
Figure 604. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 605*.

Figure 605. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

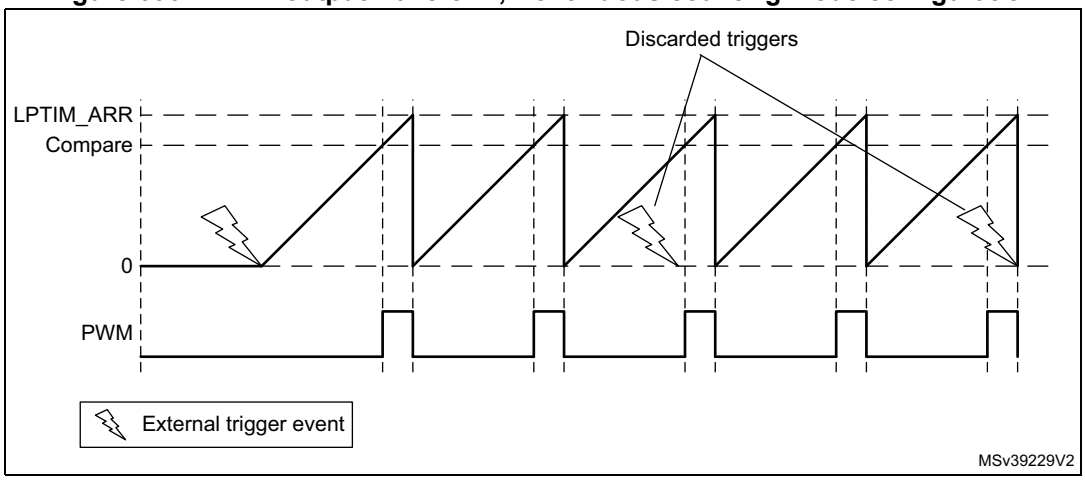
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 606*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 606. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

43.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

43.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 43.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

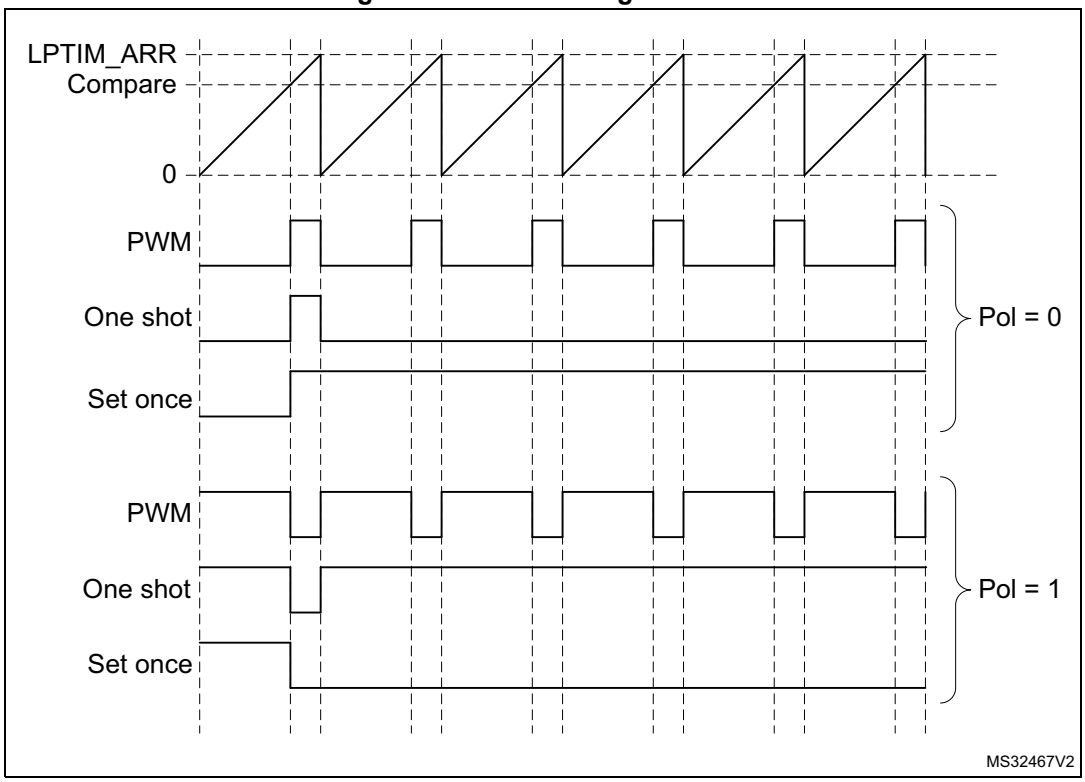
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL/CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 607* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL/CCxP bit.

Figure 607. Waveform generation



43.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

43.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don’t care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

43.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

43.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software should ensure that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

43.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

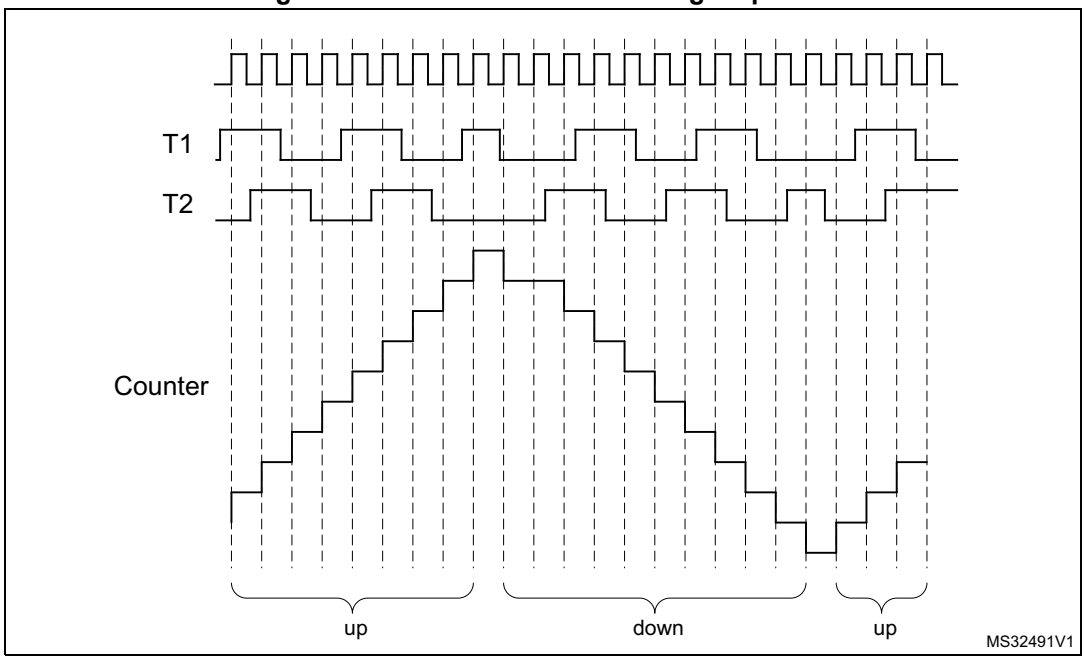
Table 479. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 608. Encoder mode counting sequence



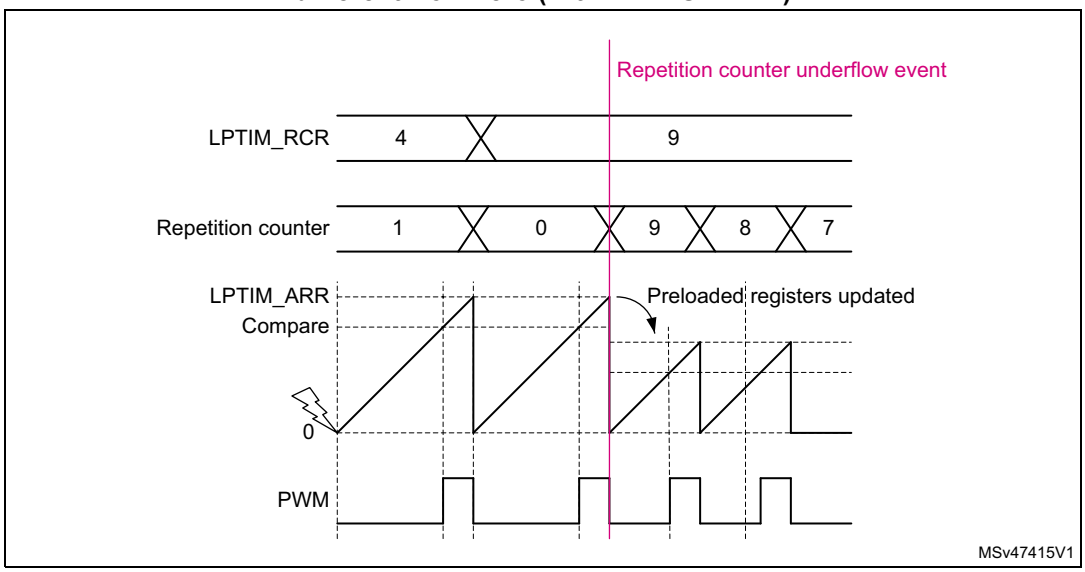
43.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 609*.

Figure 609. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 43.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

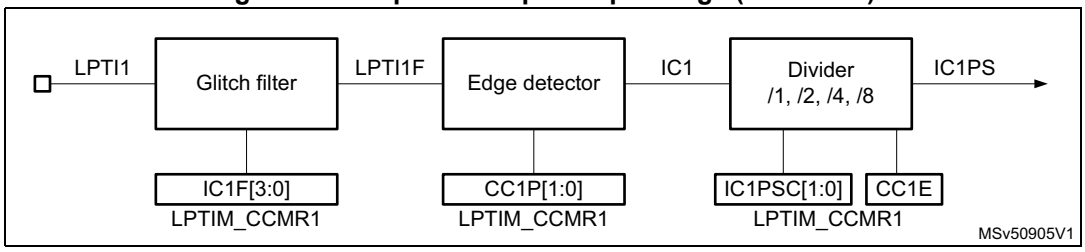
43.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

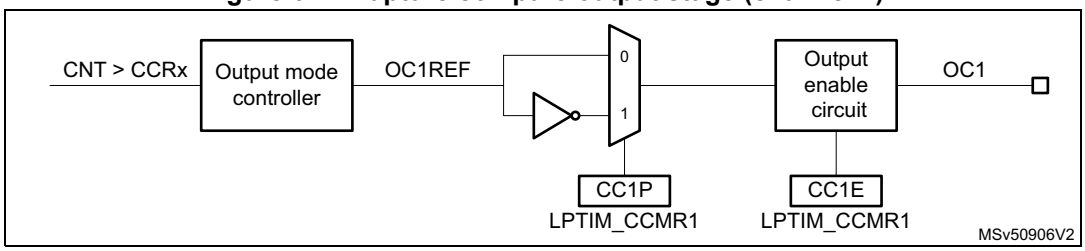
Figure 610. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 611. Capture/compare output stage (channel 1)



43.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 480*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 480*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 480. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 480. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

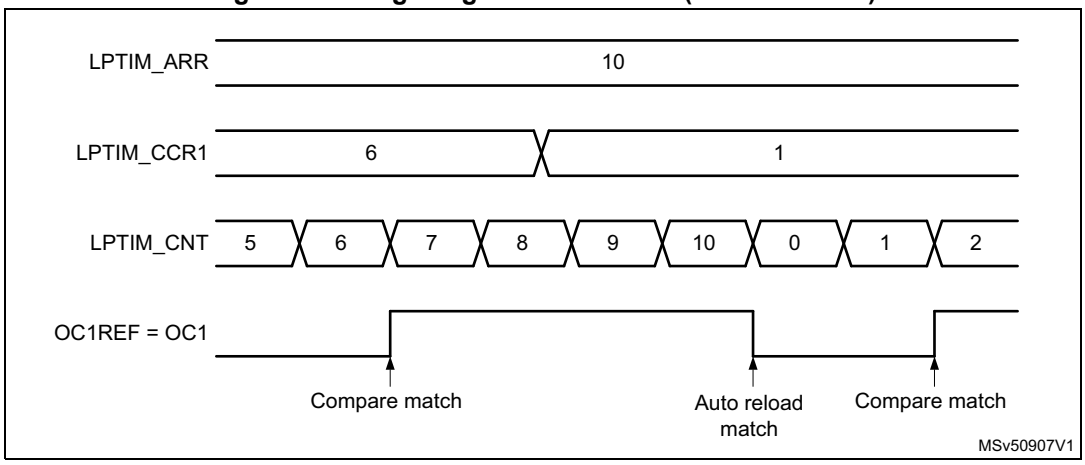
43.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 612 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

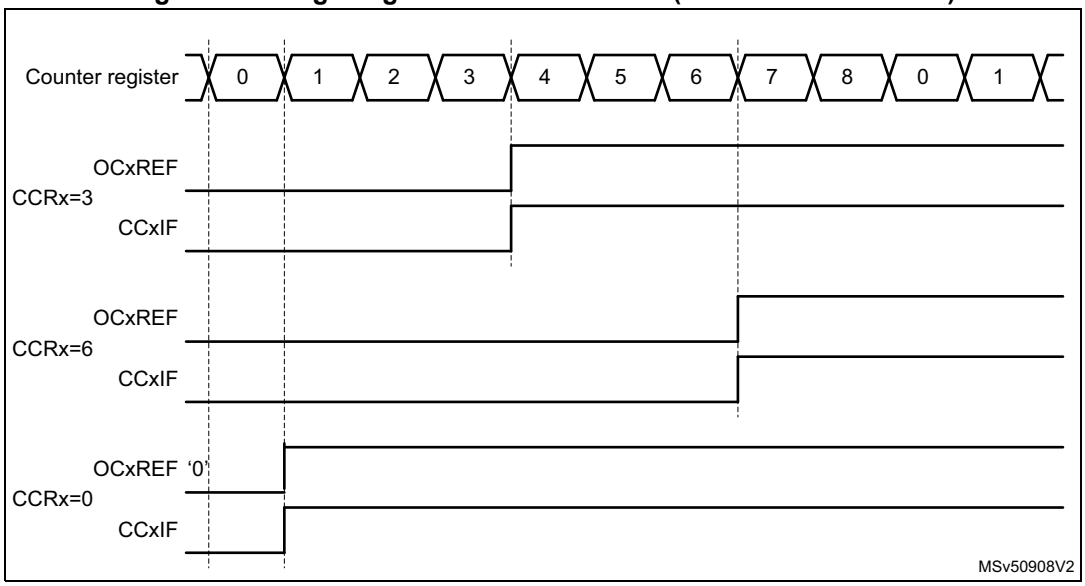
Figure 612. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 613 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 613. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

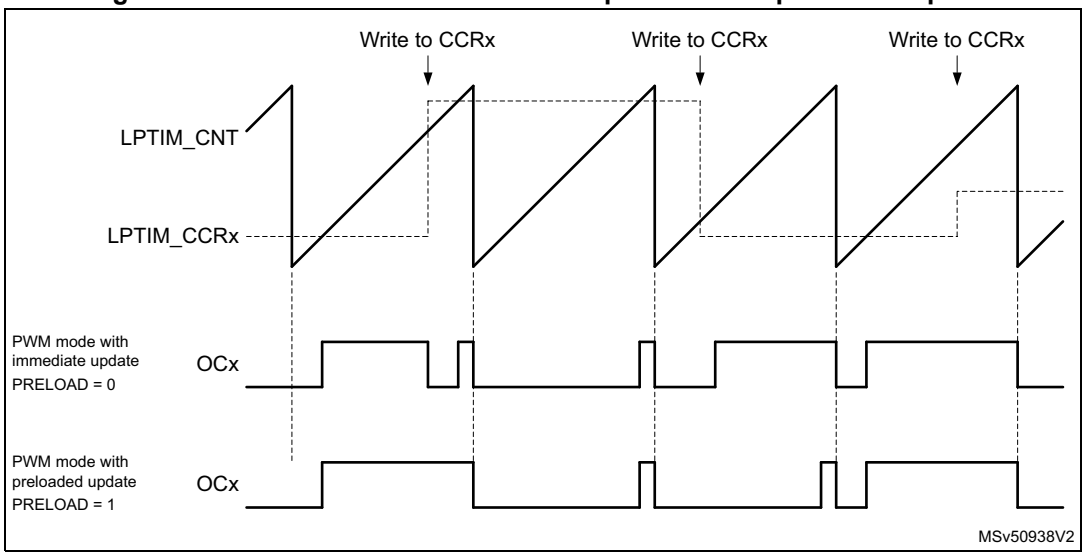
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 614* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 614. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 612 illustrates this behavior.

43.4.20 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

43.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

43.5 LPTIM low-power modes

Table 481. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

Note: All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

43.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 482. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture(2)	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture(2)	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(3)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(3)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 43.3: LPTIM implementation*.
2. If LPTIM does not implement any channel this event does not exist. Refer to *Section 43.3: LPTIM implementation*.
3. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 43.3: LPTIM implementation*.

43.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

43.7.1 LPTIM4 interrupt and status register (LPTIM4_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN	UP	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
							r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3: LPTIM implementation.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

43.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3, 5, 6)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

43.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3, 5, 6)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OF	CC1 OF	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	Res.	EXT TRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

43.7.4 LPTIM4 interrupt clear register (LPTIM4_ICR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
							w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

43.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3, 5, 6)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag
Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

43.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3, 5, 6)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OCF	CC1 OCF	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARRO KCF	Res.	EXTTR IGCF	ARRM CF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

43.7.7 LPTIM4 interrupt enable register (LPTIM4_DIER)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

43.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3, 5, 6)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

43.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3, 5, 6)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 43.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 43.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

43.7.10 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable

The ENC bit controls the Encoder mode

0: Encoder mode disabled

1: Encoder mode enabled

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 43.3.

Bit 23 **COUNTMODE**: counter mode enabled

The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:

0: The counter is incremented following each internal clock pulse

1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode

The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality

0: Registers are updated after each APB bus write access

1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity

The WAVPOL bit controls the output polarity

0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers

1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers

Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 43.3.

Bit 20 **WAVE**: Waveform shape

The WAVE bit controls the output shape

0: Deactivate Set-once mode

1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable

The TIMOUT bit controls the Timeout feature

0: A trigger event arriving when the timer is already started is ignored

1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity

The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:

00: Software trigger (counting start is initiated by software)

01: Rising edge is the active edge

10: Falling edge is the active edge

11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7

See *Section 43.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger
01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition
01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 43.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

43.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

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

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

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

43.7.12 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

If LPTIM does not implement any channel:

The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

43.7.13 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

43.7.14 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value
When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

43.7.15 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

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

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 43.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 43.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection

The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 43.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection

The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 43.4.3: LPTIM input and trigger mapping*.

43.7.16 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

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

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

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

43.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

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

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)

kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

If LPTIM does not implement any channel this register is reserved. Refer to Section 43.3.

43.7.18 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 43.3: LPTIM implementation*.

43.7.19 LPTIM register map

The following table summarizes the LPTIM registers.

Table 483. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIM4_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0																0	0	0	0	0	0	0	0	0

Table 483. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIMx_ISR (x = 1 to 3, 5, 6) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ISR (x = 1 to 3, 5, 6) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF(1)	CC1OF	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIM4_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0																0	0	0	0	0	0	0	0	0
0x004	LPTIMx_ICR (x = 1 to 3, 5, 6) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ICR (x = 1 to 3, 5, 6) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF(1)	CC1OCF	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIM4_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value																								0	0	0	0	0	0	0	0	0
0x008	LPTIMx_DIER (x = 1 to 3, 5, 6) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_DIER (x = 1 to 3, 5, 6) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE(1)	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE(1)	CC1OIE	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	WAVPOL(3)	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]		Res.	PRESC			Res.	TRGFLT		Res.	CKFLT		CKPOL		CKSEL	
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 483. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	LPTIM_ CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1(4)	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 43.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 43.3: LPTIM implementation*.
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to *Section 43.3: LPTIM implementation*.
4. If LPTIM does not implement any channel this register is reserved. Refer to *Section 43.3: LPTIM implementation*.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 43.3: LPTIM implementation*.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

44 Independent watchdog (IWDG)

44.1 IWDG introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 44.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

44.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

44.3 IWDG implementation

Table 484. IWDG features (1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation (2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate an interrupt in system Stop	X
Capability to generate an interrupt in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode (3)	X
Option bytes to control the activity in Stop mode (4)	X
Option bytes to control the activity in Standby mode (5)	X
Option bytes to control the Hardware mode (6)	X

1. 'X' = supported, '-' = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.

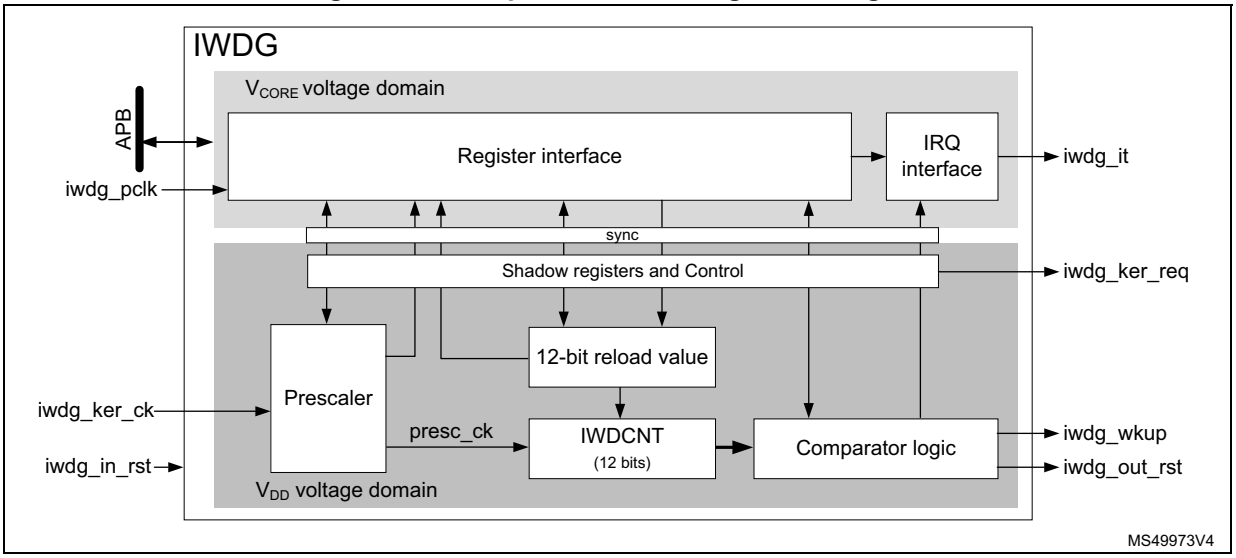
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

44.4 IWDG functional description

44.4.1 IWDG block diagram

Figure 615 shows the functional blocks of the independent watchdog module.

Figure 615. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 44.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 485. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

44.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 486*.

Table 486. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

44.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The Software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 44.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

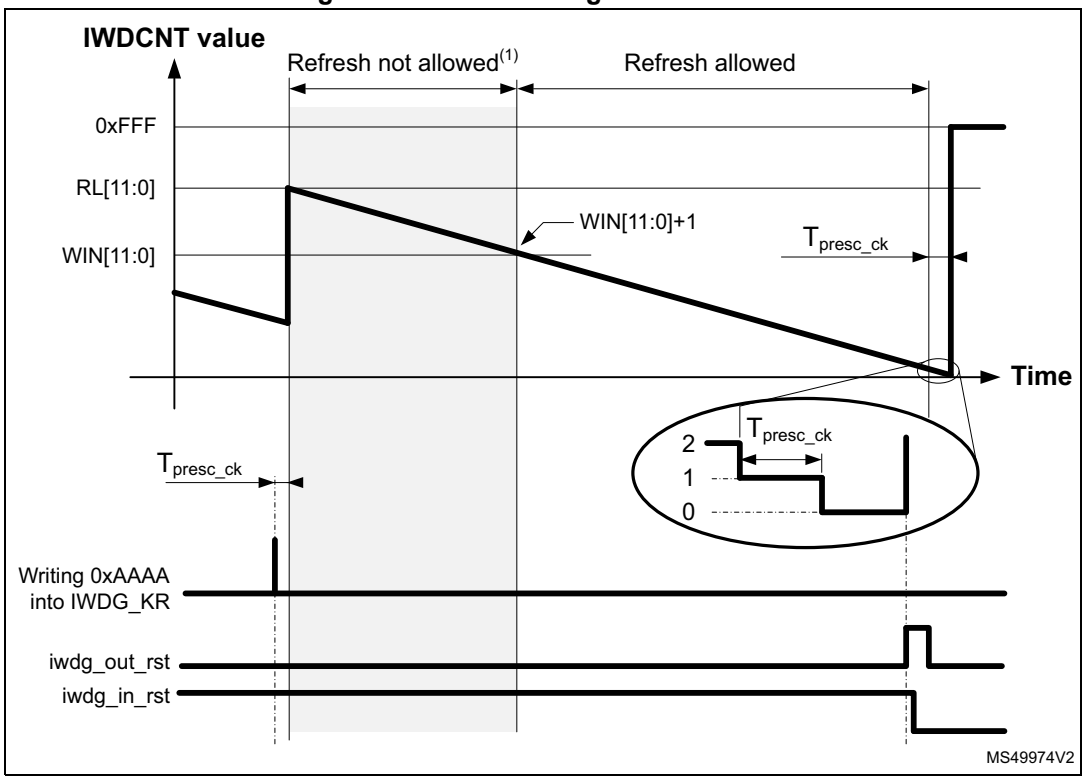
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 616*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 616. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

44.4.4 Window option

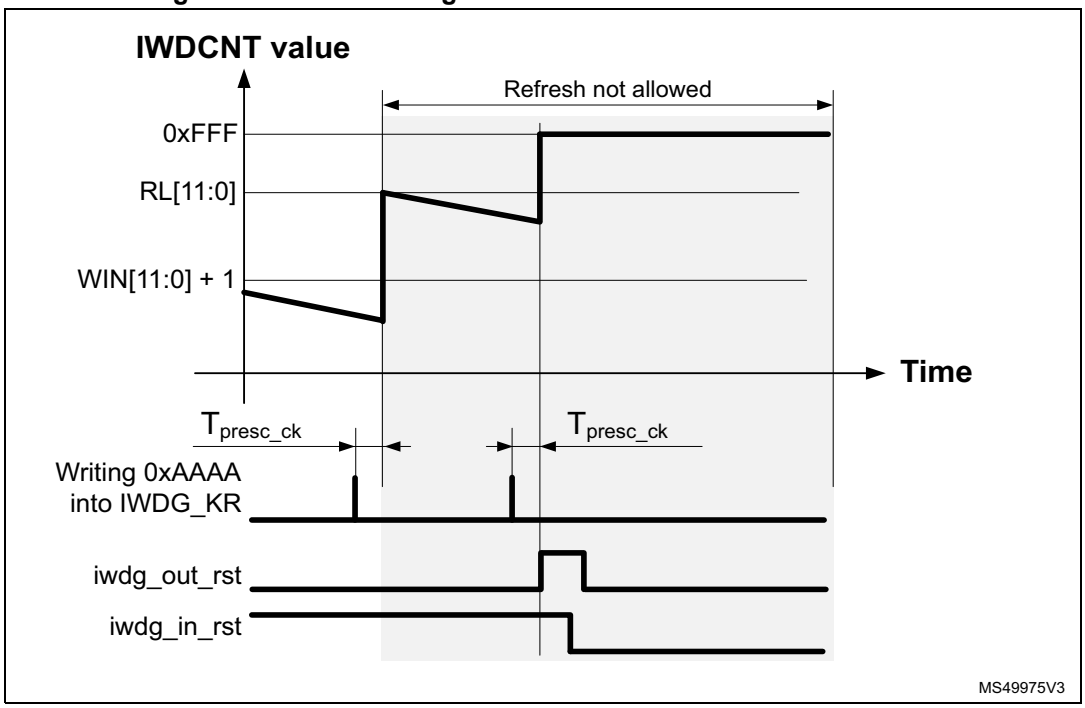
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 617*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 617. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

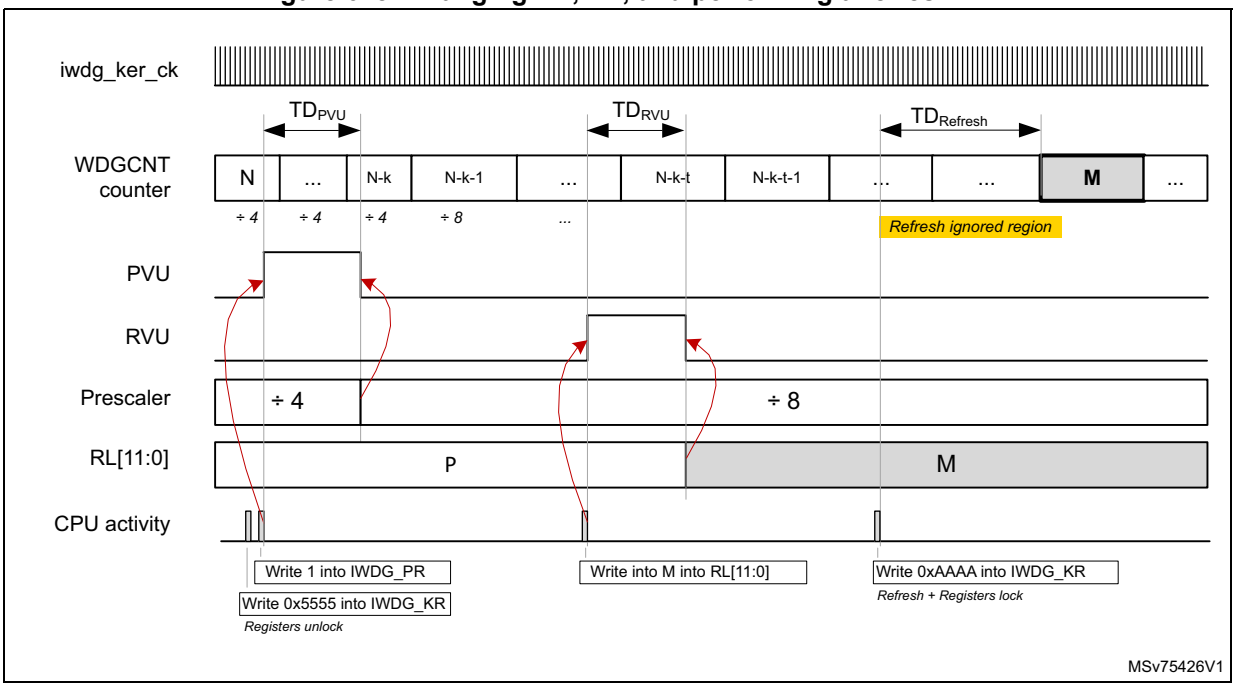
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 618. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 485: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

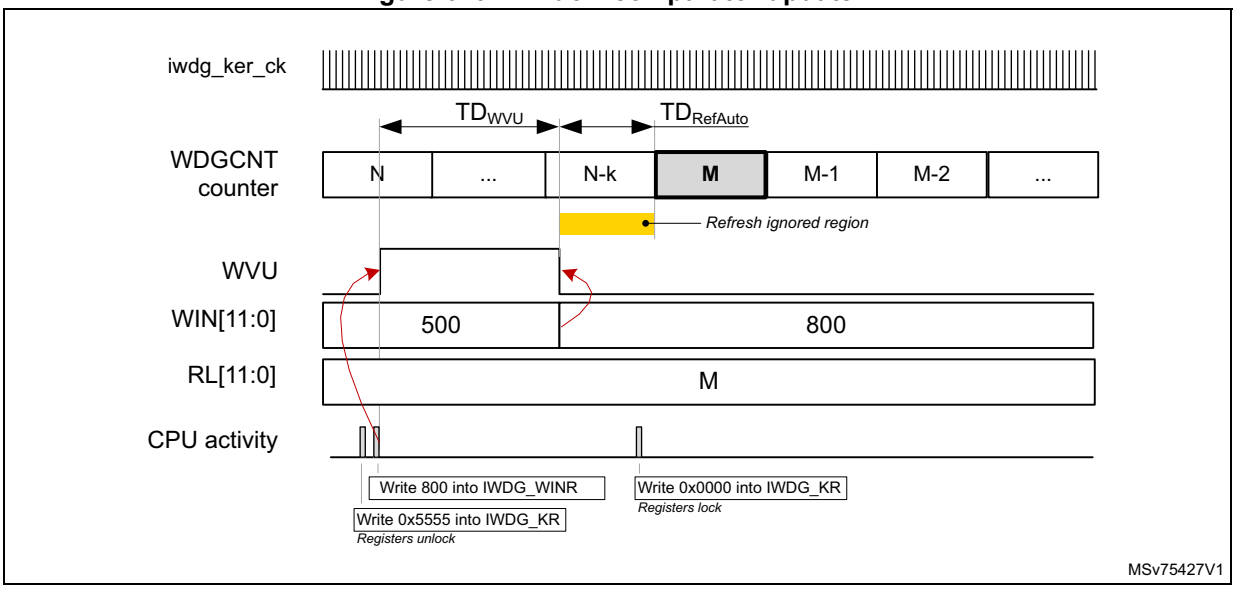
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 619 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 619. Window comparator update(1)



1. Refer to *Table 485: IWDG delays versus actions* for details on timing values.

44.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 44.3* for details on the capabilities of this product.

44.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the

sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

44.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 44.3* for details.

Table 487. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 44.3* for details. IWDG interrupts cause the device exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 44.3* for details. IWDG interrupts do not make the device to exit from Standby mode.

44.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

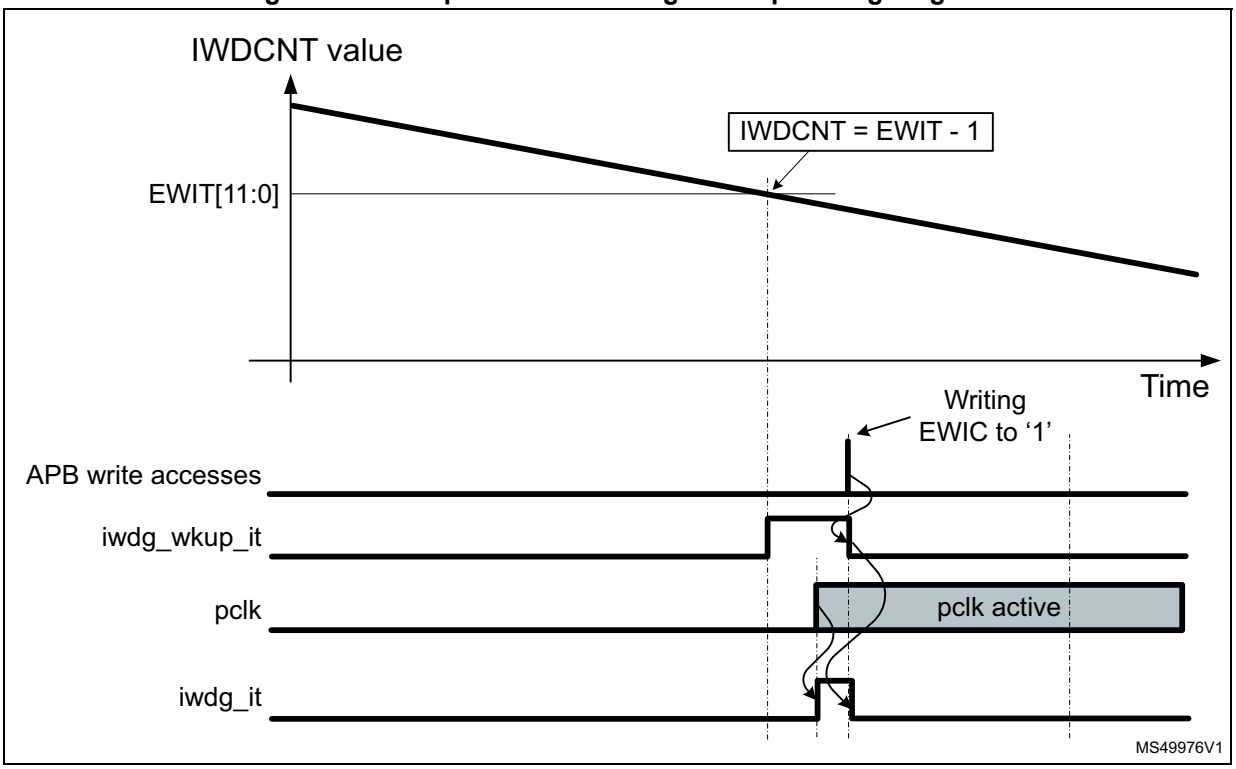
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG early wake-up interrupt register (IWDG_EWCR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 620. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

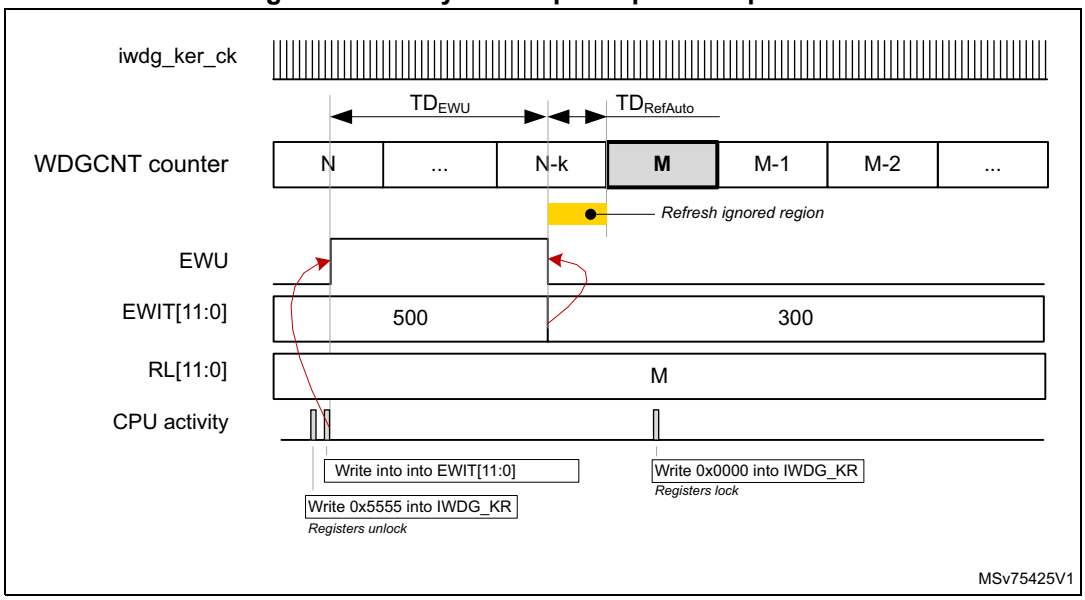
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 620 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 621. Early wake-up comparator update(1)



1. Refer to *Table 485: IWDG delays versus actions* for details on timing values.

Table 488 summarizes the IWDG interrupt request.

Table 488. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

44.7 IWDG registers

Refer to *Section 1.2 on page 104* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

44.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

44.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 44.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

44.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 44.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

44.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in

software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 485: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 485: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 485: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 485: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the

early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

44.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 44.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

44.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bit 14 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter window value

These bits are write access protected (see *Section 44.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

44.7.7 IWDG register map

Table 489. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													0	0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																		0						x					0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	EWIC	Res.	Res.	EWIT[11:0]											
	Reset value																	0	0			0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

45 System window watchdog (WWDG)

45.1 WWDG introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

45.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 623*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

45.3 WWDG implementation

Table 490. WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the hardware mode(4)	X

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

45.4 WWDG functional description

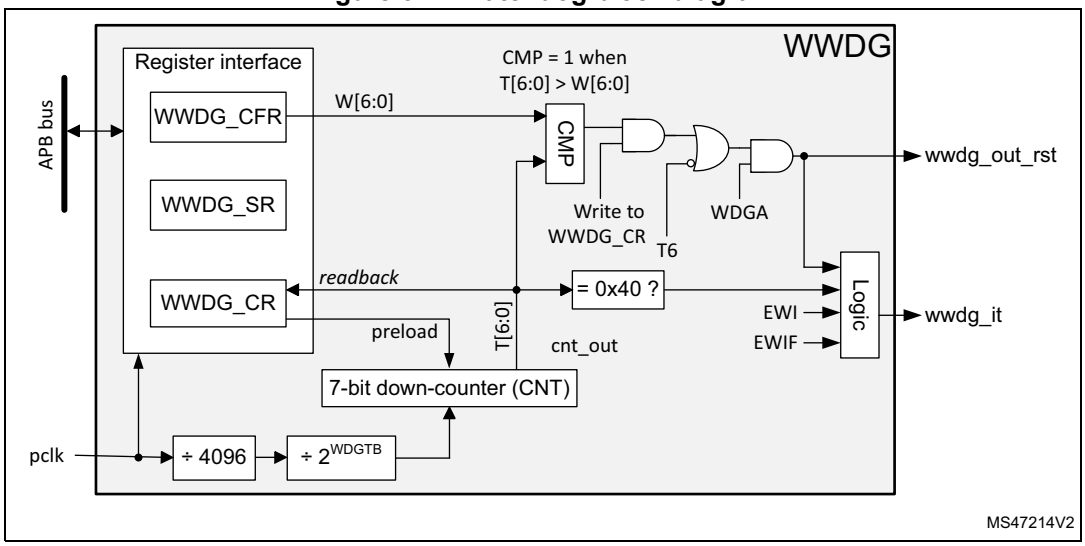
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 622* for the WWDG block diagram.

45.4.1 WWDG block diagram

Figure 622. Watchdog block diagram



45.4.2 WWDG internal signals

Table 491 gives the list of WWDG internal signals.

Table 491. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

45.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

45.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 623*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 623* describes the window watchdog process.

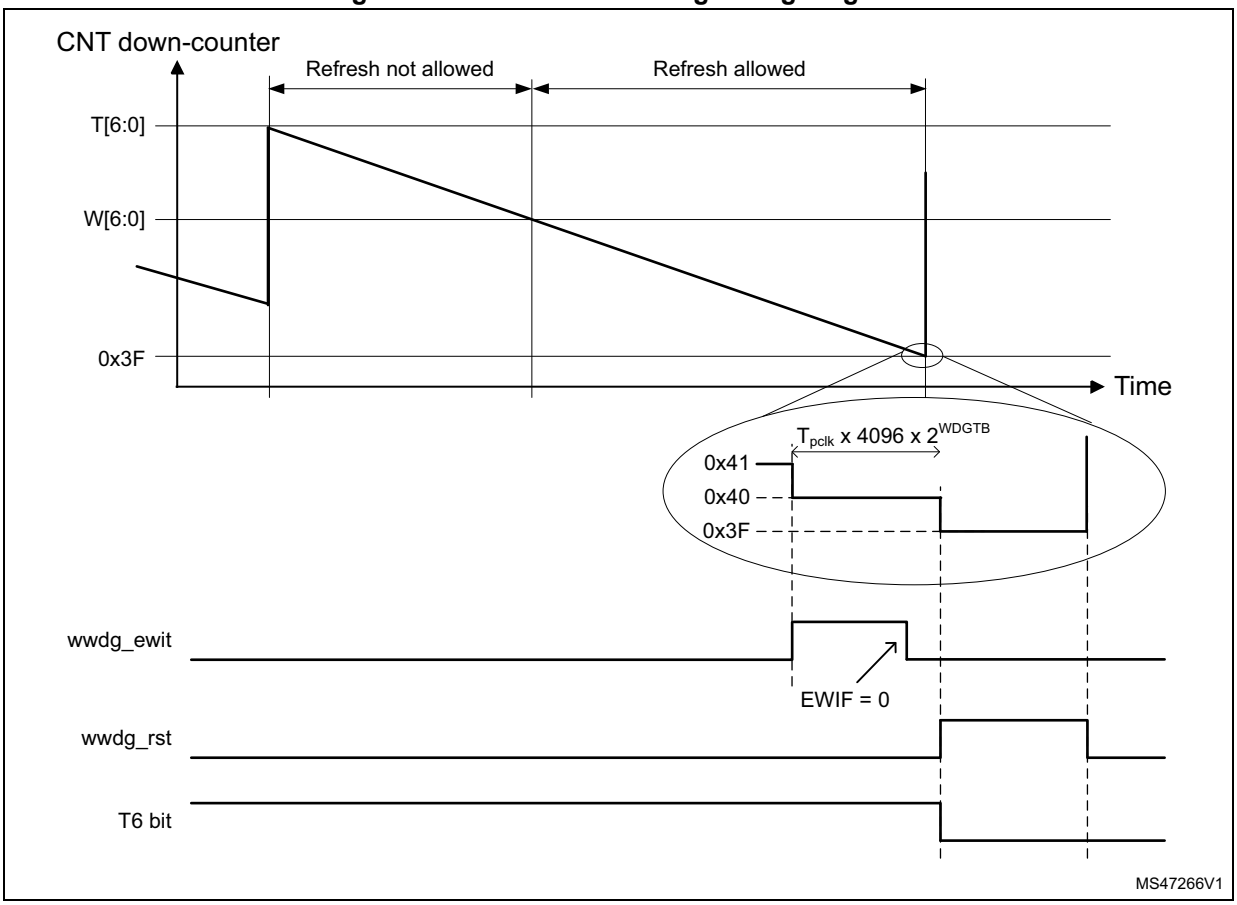
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

45.4.5 How to program the watchdog timeout

Use the formula in *Figure 623* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 623. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

45.4.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 59: Debug support (DBG)*.

45.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

Table 492. WWDG interrupt requests

Interrupt		Event flag	Enable control bit	Interrupt clearing method	Exit from mode		
Vector	Event				Sleep	Stop(1)	Standby(1)
WWDG(2)	Early wake-up interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 45.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

45.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

45.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

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

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

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

45.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

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

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

45.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

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

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

45.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 493. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

46 Real-time clock (RTC)

46.1 RTC ntroduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in V_{BAT} mode.

46.2 RTC main features

The RTC supports the following features (see *Figure 624: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the VBAT pin.

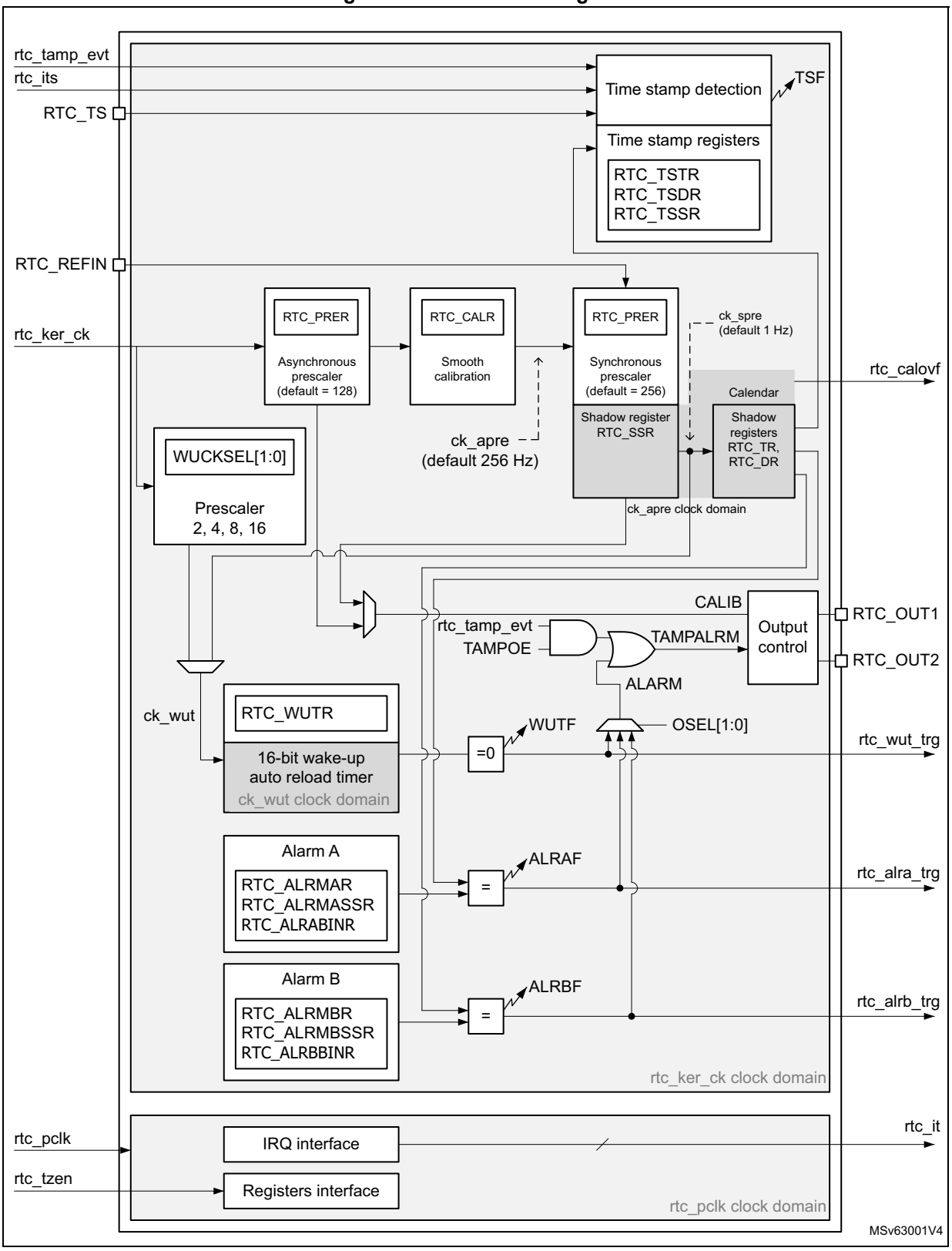
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

46.3 RTC functional description

46.3.1 RTC block diagram

Figure 624. RTC block diagram



46.3.2 RTC pins and internal signals

Table 494. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 495. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_tzen	Input	RTC TrustZone enabled
rtc_it	Output	RTC interrupts (refer to *Section 46.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 46.4: RTC low-power modes* for more details.

Table 496. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_tzen	From FLASH option bytes: TZEN
rtc_calovf	To TAMP peripheral: tamp_itamp5

The TZEN option bit is used to activate TrustZone in the device.

TZEN = 1: TrustZone activated.

TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB access to the RTC registers are nonsecure.

The triggers outputs can be used as triggers for other peripherals.

46.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 and TAMP_OUT3 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in *Table 497*.

Table 497. RTC pin PC13 configuration(1)

PC13 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP3E=TAMP3AM=1 with ATOSHARE=0 and OUT3_RMP=00, or TAMPxE=TAMPxAM=1 with ATOSHARE=1, ATOSELx=2 and OUT3_RMP=00	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care	Don't care	Don't care
		00	1								
		01 or 10 or 11	1								
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care	Don't care	Don't care
		00	1								
		01 or 10 or 11	1								
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care	Don't care	Don't care
		00	1								
		01 or 10 or 11	1								
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT2 output PP		00	0	0	0	Don't care	Don't care	1	0	Don't care	Don't care
TAMP_OUT3 output PP		00	0	0	0	Don't care	Don't care	0	1	Don't care	Don't care

Table 497. RTC pin PC13 configuration(1) (continued)

PC13 Pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP3E=TAMP3AM=1 with ATOSHARE=0 and OUT3_RMP=00, or TAMPxE=TAMPxAM=1 with ATOSHARE=1, ATOSELx=2 and OUT3_RMP=00	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMP_IN1 input floating	00	0	0	Don't care	Don't care	Don't care	0	0	1	0
	00	0	1	1						
	Don't care	Don't care	0							
RTC_TS and TAMP_IN1 input floating	00	0	0	Don't care	Don't care	Don't care	0	0	1	1
	00	0	1	1			0	0		
	Don't care	Don't care	0				0	0		
RTC_TS input floating	00	0	0	Don't care	Don't care	Don't care	0	0	0	1
	00	0	1	1			0	0		
	Don't care	Don't care	0				0	0		
Wakeup pin	00	0	0	Don't care	Don't care	Don't care	0	0	0	0
	00	0	1	1			0	0		
	Don't care	Don't care	0				0	0		
Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0	0	0
	00	0	1	1			0	0		
	Don't care	Don't care	0				0	0		

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

RTC_OUT2, TAMP_IN2 and TAMP_OUT3 are mapped on the same pin (PI8). PI8 configuration is controlled by the RTC, whatever the PI8 GPIO configuration. The RTC or TAMP functions mapped on PI8 are available in all low-power modes and in VBAT mode.

The output mechanism follows the priority order shown in *Table 498*.

Table 498. PI8 configuration

PI1 Pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN=1 and OUT2_RMP=0(1)	TAMPALRM_TYPE	TAMPALRM_PU	TAMP3E=TAMP3AM=1 with ATOSHARE=0 and OUT3_RMP=01, or TAMPxE=TAMPxAM=1 with ATOSHARE=1, ATOSELx=2 and OUT3_RMP=01	TAMP2E & IN2_RMP
TAMPALRM output Push-Pull		01 or 10 or 11	0	Don't care	1	0	0	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
TAMPALRM output Open-Drain	No pull	01 or 10 or 11	0	Don't care	1	1	0	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
	Internal pull-up	01 or 10 or 11	0	Don't care	1	1	1	Don't care	Don't care
		00	1						
		01 or 10 or 11	1						
CALIB output PP		00	0	1	1	Don't care	Don't care	Don't care	Don't care
TAMP_OUT3 output PP		00	0	0	0	Don't care	Don't care	1	0
TAMP_IN2 input floating		00	0	0	0	Don't care	Don't care	0	1
Wakeup pin		00	0	0	0	Don't care	Don't care	0	0
Standard GPIO		00	0	0	0	Don't care	Don't care	0	0

1. RTC_OUT2 is mapped on PI8 when OUT2_RMP = 0.

In addition, it is possible to output RTC_OUT2 on PB2 or PI8 pin thanks to OUT2EN and OUT2_RMP bits. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table *Table 499*.

Table 499. RTC_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PB2 or PI8(1)
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

1. RTC_OUT2 is mapped on PI8 when OUT2_RMP = 0, and on PB2 when OUT2_RMP = 1.

46.3.4 RTC secure protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes, except for the RTC secure configuration register (RTC_SECCFGR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When the SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC control bits. If all

these bits are also clear, all the RTC registers can be read and written in secure and nonsecure mode.

- When INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.
  - ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCFGR can be written only in secure mode.
- When WUTSEC is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When TSSEC is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_SMISR can be read and written only in secure mode.
  - TSPRIV in the RTC_PRIVCFGR can be written only in secure mode

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- In case the register has a global protection: a notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller). In case only a few bits of the register are protected (for registers with mixed features such as RTC_CR...), no notification is generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

46.3.5 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and non-privileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and non-privilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and non-privilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRASEC in the RTC_SECCFGR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE, ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRBSEC in the RTC_SECCFGR can be written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTSEC in the RTC_SECCFGR can be written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the

RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.

– TSSEC in the RTC_SECCFGR can be written only in privilege mode.

A non-privileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

46.3.6 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ? 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 624: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 46.3.10: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

46.3.7 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 46.6.12: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

46.3.8 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

46.3.9 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

46.3.10 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 1980*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ? WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to *Section 46.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop, and Standby) have no influence on the wake-up timer.

46.3.11 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

The registers protected by INITPRIV are write-protected by the INIT KEY.

The registers protected by CALPRIV are write-protected by the CAL KEY.

In case PRIV or INITPRIV is set in the RTC_PRIVCFGR, and/or SEC or INITSEC is set in the RTC_SECCFGR: the INIT KEY is unlocked and locked only if the write accesses into

the RTC_WPR register are done in the privilege and security mode defined by PRIV, INITPRIV, SEC, INITSEC configuration.

In case PRIV or CALPRIV is set in the RTC_PRIVCFGR, and/or SEC or CALSEC is set in the RTC_SECCFGR: the CAL KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege and security mode defined by PRIV, CALPRIV, SEC, CALSEC configuration.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   – If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   – If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   – If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   – If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 ?s with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

46.3.12 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third

read access must be done. In any case the APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 1979*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 46.3.14: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

46.3.13 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers

(RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

46.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

46.3.15 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

46.3.16 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

46.3.17 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and refer to *Section : Timestamp on tamper event*.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs, due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

46.3.18 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6^{th} stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8^{th} stage of the synchronous prescaler.

46.3.19 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ? 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

46.4 RTC low-power modes

Table 500. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 501. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby mode	Functional in Standby mode	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	Yes	Yes

46.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The nonsecure interrupt output or the secure interrupt output is also activated.

Table 502. Nonsecure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE and (ALRASEC=0 and SEC=0)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=0 and SEC=0)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=0 and SEC=0)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=0 and SEC=0)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=0 and SEC=0)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

Table 503. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC_S	Alarm A	ALRAF	ALRAIE and (ALRASEC=1 or SEC=1)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=1 or SEC=1)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=1 or SEC=1)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=1 or SEC=1)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=1 or SEC=1)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_SMISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

46.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

46.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1979* and *Reading the calendar on page 1980*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1978*.

This register can be write-protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

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

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

46.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 1979* and *Reading the calendar on page 1980*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1978*.

This register can be write-protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

46.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

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

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

46.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1978*.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

It is cleared by hardware in initialization mode.

0: Wake-up timer configuration update not allowed except in initialization mode

1: Wake-up timer configuration update allowed

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

46.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 1979*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 1978*.

This register can be write-protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

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

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

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

46.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

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

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

46.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 1978.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ? 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ? 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ? 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.
1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM
0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event
0: Tamper detection event does not cause a RTC timestamp to be saved
1: Save RTC timestamp on tamper detection event
TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.
Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 **ITSE**: timestamp on internal event enable
0: internal event timestamp disabled
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable
This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wake-up output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 46.3.18: Calibration clock output*.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.

0: No effect

1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

0: Timestamp interrupt disable

1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable

0: Wake-up timer interrupt disabled

1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable

0: Alarm B interrupt disable

1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable

0: Alarm A interrupt disabled

1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable

0: timestamp disable

1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable

0: Wake-up timer disabled

1: Wake-up timer enabled

Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable

0: Alarm B disabled

1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable

0: Alarm A disabled

1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable

0: SSR underflow interrupt disabled

1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format

0: 24 hour/day format

1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers

0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.

1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.

Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

46.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the RTC_SECCFGR configuration (refer to *Section 46.3.5: RTC privilege protection modes*).

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection
0: All RTC registers can be written when the APB access is privileged or non-privileged, except the registers protected by other privilege protection bits.
1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection
0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or non-privileged.
1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, Delight saving, calibration and reference clock privilege protection
0: Shift register, Delight saving, calibration and reference clock can be written when the APB access is privileged or non-privileged.
1: Shift register, Delight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection
0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection
0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection
0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection
0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 46.3.5: RTC privilege protection modes for details on the read protection.

46.6.9 RTC secure configuration register (RTC_SECCFGR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 46.3.5: RTC privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC**: RTC global protection

0: All RTC registers can be written when the APB access is secure or nonsecure, except the registers protected by other secure protection bits.

1: All RTC registers can be written only when the APB access is secure.

Bit 14 **INITSEC**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or nonsecure.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 **CALSEC**: Shift register, daylight saving, calibration and reference clock protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or nonsecure.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSSEC**: Timestamp protection

0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wake-up timer protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection

0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection

0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

Note: Refer to Section 46.3.4: RTC secure protection modes for details on the read protection.

46.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

46.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1978*.

This register can be write-protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be write-protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.
Refer to *Section 46.3.16: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.
Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 46.3.16: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.
Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 46.3.16: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 46.3.16: RTC smooth digital calibration on page 1983*.

46.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 1978*.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

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

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

46.6.13 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

46.6.14 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

46.6.15 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

46.6.16 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don't care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don't care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don't care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don't care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

46.6.17 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

46.6.18 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don't care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don't care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don't care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don't care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

46.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).
1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.
2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.
...
31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.
From 32 to 63: All 32 SS bits are compared and must match to activate alarm.
Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value
This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.
This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.
Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

46.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag
This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.
Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload.

Bit 5 **ITSF**: Internal timestamp flag
This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

46.6.21 RTC nonsecure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow nonsecure masked flag

This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp nonsecure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp nonsecure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow nonsecure masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp nonsecure masked flag
This flag is set by hardware when a timestamp nonsecure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer nonsecure masked flag
This flag is set by hardware when the wake-up timer nonsecure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B nonsecure masked flag
This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A nonsecure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

46.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x58

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer interrupt secure masked flag
This flag is set by hardware when the wake-up timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

46.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing '1' in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wake-up timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

46.6.24 RTC option register (RTC_OR)

Address offset: 0x60

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT2_RMP
															rw

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

Bit 0 **OUT2_RMP**: RTC_OUT2 mapping
0: RTC_OUT2 is mapped on PI8
1: RTC_OUT2 is mapped on PB2

46.6.25 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

46.6.26 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against nonsecure access. Refer to *Section 46.3.4: RTC secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 46.3.5: RTC privilege protection modes*.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.

46.6.27 RTC register map

Table 504. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x20	RTC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	INITSEC	CALSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSSEC	WUTSEC	ALRBSEC	ALRASEC
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 504. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0	0	0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0	0	0	0	0	0	0
0x60	RTC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUT2_RMP
	Reset value																																0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 504. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

47 Tamper and backup registers (TAMP)

47.1 TAMP introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 11 tamper pins and 13 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

47.2 TAMP main features

- A tamper detection can optionally erase the backup registers, backup SRAM, SRAM2, caches, and cryptographic peripherals. The device resources protected by tamper are named “device secrets”.
- 32 32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- Up to 11 tamper pins for 8 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Flexible active tamper I/O management: from 5 meshes (each input associated to its own exclusive output) to 8 meshes (single output shared for up to 8 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 13 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration.
  - Backup registers configuration in 3 configurable-size areas:

    1 read/write secure area.

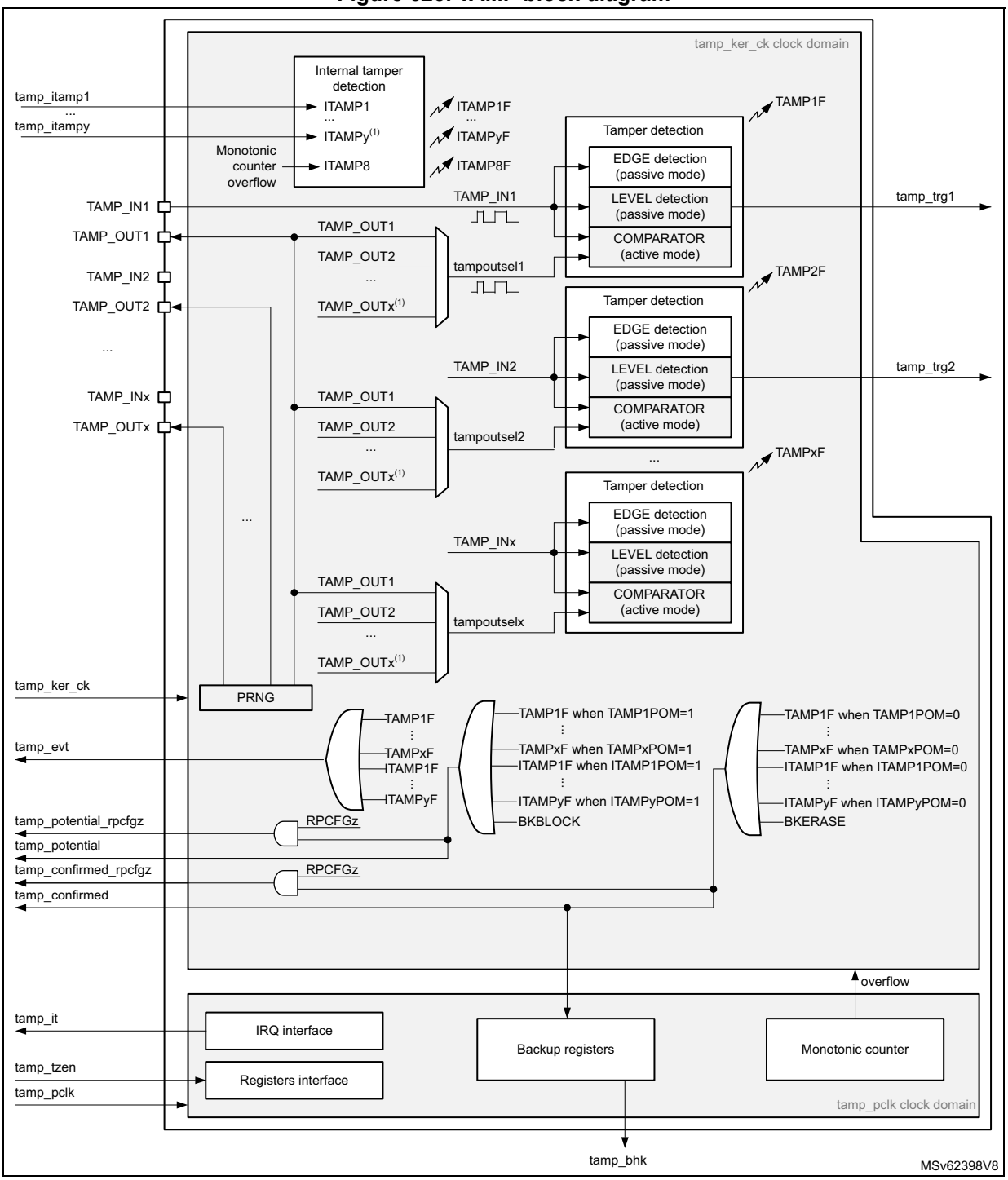
    1 write secure/read nonsecure area.

    1 read/write nonsecure area.
  - Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.

47.3 TAMP functional description

47.3.1 TAMP block diagram

Figure 625. TAMP block diagram



1. The number of external and internal tampers depends on products.

47.3.2 TAMP pins and internal signals

Table 505. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin (active mode only)

Table 506. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_tzen	Input	TAMP TrustZone enabled
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1

Table 506. TAMP internal input/output signals (continued)

Internal signal name	Signal type	Description
tamp_confirmed_rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 47.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger
tamp_bhk	Output	Tamper boot hardware key bus

1. Refer to *Table 507: TAMP interconnection*.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). The TAMP kernel clock is required only for external tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock.

Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes or V_{BAT} depending on the selected clock (refer to *Section 47.4: TAMP low-power modes* for more details.

Table 507. TAMP interconnection

Signal name	Source/Destination
tamp_tzen	From FLASH option bytes: TZEN
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_potential	The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: – backup registers – SRAM2 RHUK (root hardware unique key) in system Flash memory and BHK (boot hardware key) hardware buses to SAES are blocked. The tamp_potential signal is used to erase the device secrets listed hereafter: – ICACHE content – SAES, AES, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is ongoing.

Table 507. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_confirmed	The tamp_confirmed signal is used to erase the device secrets listed hereafter: – backup registers – SRAM2 – ICACHE/DCACHE content – OTFDEC keys and CRC registers – SAES, AES, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is ongoing. RHUK in system Flash memory (root hardware unique key) hardware bus to SAES is blocked.
tamp_potential_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg0 signal is used to block the read and write accesses to the device secrets listed hereafter: – Backup SRAM
tamp_confirmed_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg0 signal is used to erase the device secrets listed hereafter: – Backup SRAM The device secrets access is blocked when erase is on-going.
tamp_itamp1	Backup domain voltage threshold monitoring(1)
tamp_itamp2	Temperature monitoring(1)
tamp_itamp3	LSE monitoring (LSECSS)(2)
tamp_itamp4	HSE monitoring (rcc_hsecss_fail)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG/SWD access when NVSTATE ? OPEN
tamp_itamp7	ADC2 watchdog monitoring 1
tamp_itamp8(3)	Monotonic counter 1 overflow
tamp_itamp9	Cryptographic peripherals fault (SAES or AES or PKA or TRNG)
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_itamp12	ADC2 analog watchdog monitoring 2
tamp_itamp13	ADC2 analog watchdog monitoring 3
tamp_itamp15	System fault
tamp_bhk	saes_bhk. This bus is used to load the boot hardware key in the secure AES co-processor.

1. This monitoring must be enabled by setting MONEN in *PWR Backup domain control register (PWR_BDCR)*.
2. This monitoring must be enabled by setting LSEON in *PWR Backup domain control register (PWR_BDCR)*.
3. This signal is generated in the TAMP peripheral.

The TZEN option bit is used to activate TrustZone in the device.

TZEN = 1: TrustZone activated.

TZEN = 0: TrustZone disabled.

When TrustZone is disabled, the APB access to the TAMP registers are nonsecure.

47.3.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 46.3.3: GPIOs controlled by the RTC and TAMP*.

47.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

47.3.5 TAMP secure protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes, except for the TAMP secure configuration register (TAMP_SECCFGR) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in the TAMP_SECCFGR register:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR is always possible in secure and nonsecure modes. All the other TAMP registers can be read only in secure mode, except for the backup registers and monotonic counters which have their own protection setting.
- When the CNT1SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT1R can be read and written only in secure mode.

A nonsecure access to a secure-protected register is denied:

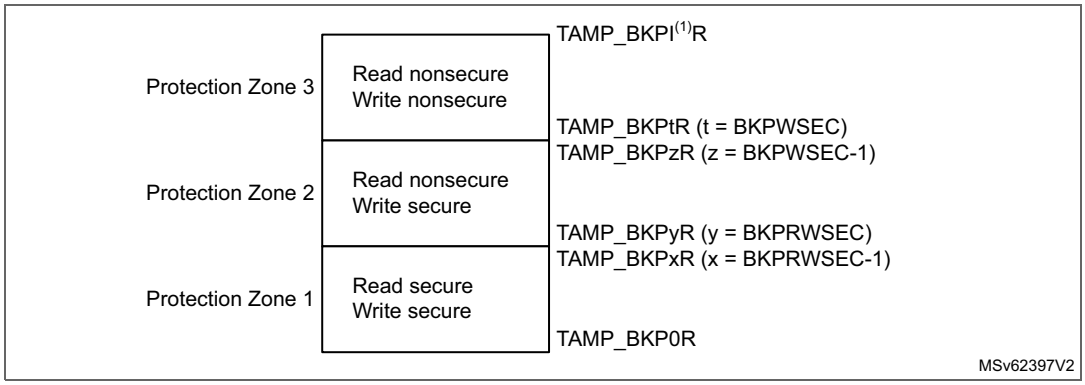
- There is no bus error generated.
- A notification is generated through a flag/interrupt in the TZIC (TrustZone illegal access controller).
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

47.3.6 Backup registers protection zones

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below):

Figure 626. Backup registers protection zones



1. I = last backup register index

In case TZEN =1, the bits BKPWPRIV and BKPRWPRIV in the TAMP_PRIVCFGR can be written only in secure mode.

47.3.7 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and non-privileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:

- Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counter which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode.
- Reading TAMP_SECCFGR, TAMP_PRIVCFGR is always possible in privilege and non-privilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counter which have their own protection setting.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 626*). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

A non-privileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

47.3.8 Boot hardware key (BHK)

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.

For this purpose, these registers must belong to the Protection Zone 1: BKPRWSEC must be greater or equal to 8.

Once the backup registers are written with the boot hardware key, the BHKLOCK bit must be set in the TAMP_SECCFGR register. Once BHKLOCK is set, the 8 backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a tamper event or when the product state is opened. It is also cleared with BKERASE command (in all cases the backup registers are also erased).

Refer to section secure AES co-processor (SAES) for details on procedure to download the boot hardware key in the SAES.

47.3.9 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 507: TAMP interconnection*. The device secrets list is configurable thanks to *TAMP resources protection configuration register (TAMP_RPCFGR)*.
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 507: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:

- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

47.3.10 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Note: The backup registers are also erased when the product state is changed from Closed to Open.

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the *Table 507: TAMP interconnection*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the *Table 507: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 507: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 507: TAMP interconnection*
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 507: TAMP interconnection*

Table 508. Device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 507: TAMP interconnection*.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

47.3.11 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ? 00 and active mode is not selected). Refer to *Section : Level detection with filtering on tamper inputs (passive mode)*.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 627. Tamper sampling with precharge pulse

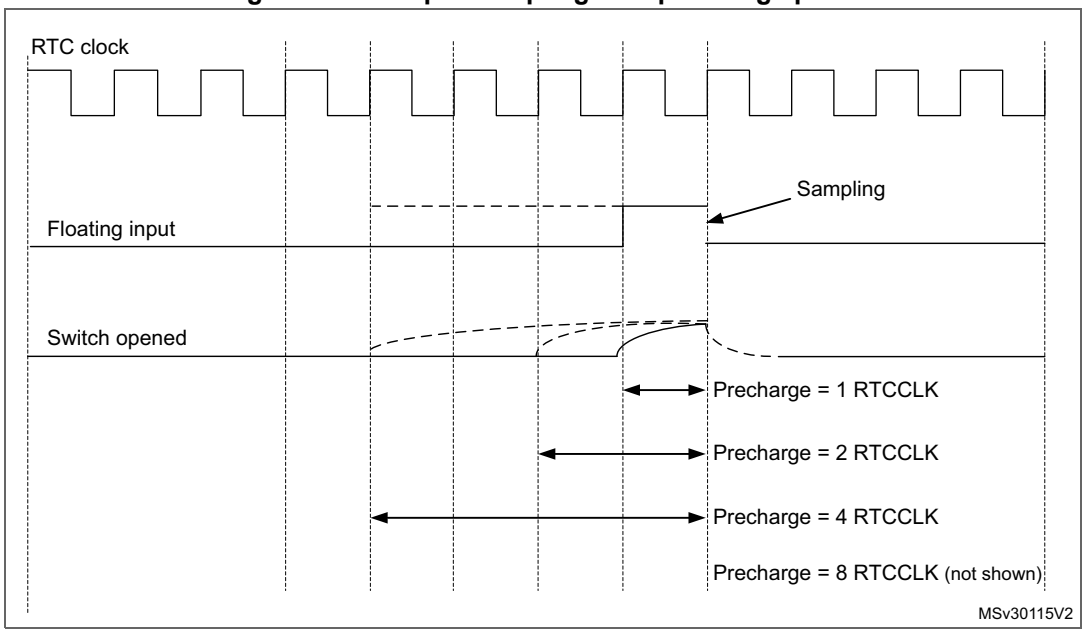
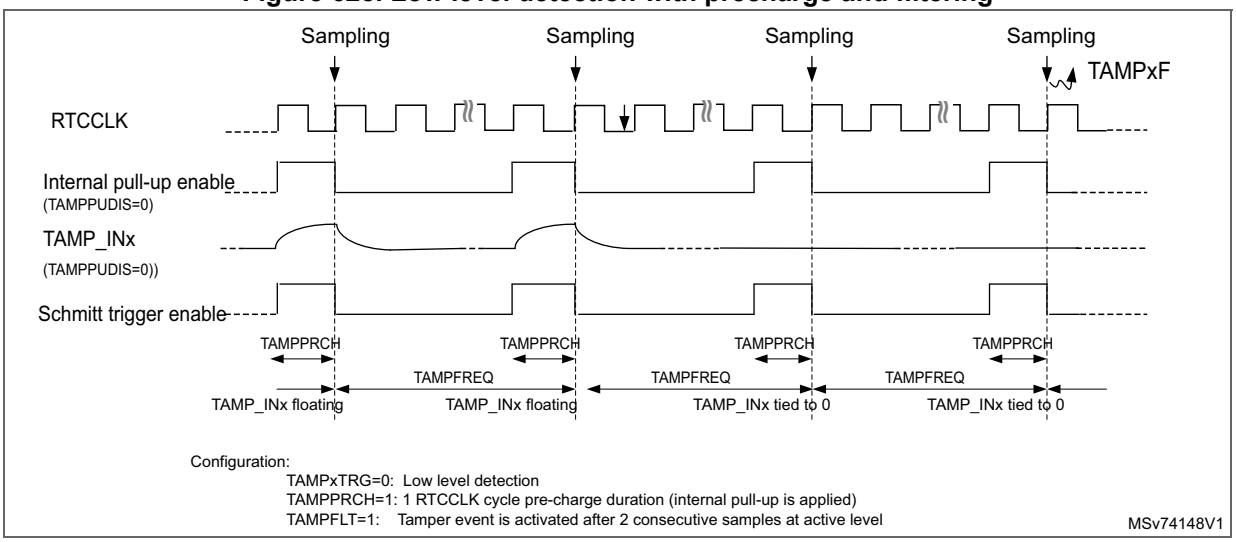


Figure 628. Low level detection with precharge and filtering



Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to

ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

Table 509. Active tamper output change period

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x0	f_{RTCCLK}	0x0	f_{RTCCLK}	0.030
		0x1	$f_{RTCCLK}/2$	0.061
		0x2	$f_{RTCCLK}/4$	0.122
		0x3	$f_{RTCCLK}/8$	0.244
		0x4	$f_{RTCCLK}/16$	0.488
		0x5	$f_{RTCCLK}/32$	0.977
		0x6	$f_{RTCCLK}/64$	1.953
		0x7	$f_{RTCCLK}/128$	3.906
…	…	…	…	…
0x7	$f_{RTCCLK}/128$	0x0	$f_{RTCCLK}/128$	3.906
		0x1	$f_{RTCCLK}/256$	7.8125
		0x2	$f_{RTCCLK}/512$	15.625
		0x3	$f_{RTCCLK}/1024$	31.250
		0x4	$f_{RTCCLK}/2048$	62.5
		0x5	$f_{RTCCLK}/4096$	125
		0x6	$f_{RTCCLK}/8192$	250
		0x7	$f_{RTCCLK}/16384$	500
0xB	$f_{RTCCLK}/2048$(2)	0x0	$f_{RTCCLK}/2048$	62.5
		0x1	$f_{RTCCLK}/4096$	125
		0x2	$f_{RTCCLK}/8192$	250
		0x3	$f_{RTCCLK}/16384$	500
		0x4	$f_{RTCCLK}/32768$	1000
		0x5	$f_{RTCCLK}/65536$	2000
		0x6	$f_{RTCCLK}/131072$	4000
		0x7	$f_{RTCCLK}/262144$	8000

1. Assuming f_{RTCCLK} = 32768 Hz.
2. This setting requires that (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.

PRNG is consumed by the selected tamper outputs at a different frequency depending on

the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 510. Minimum ATPER value

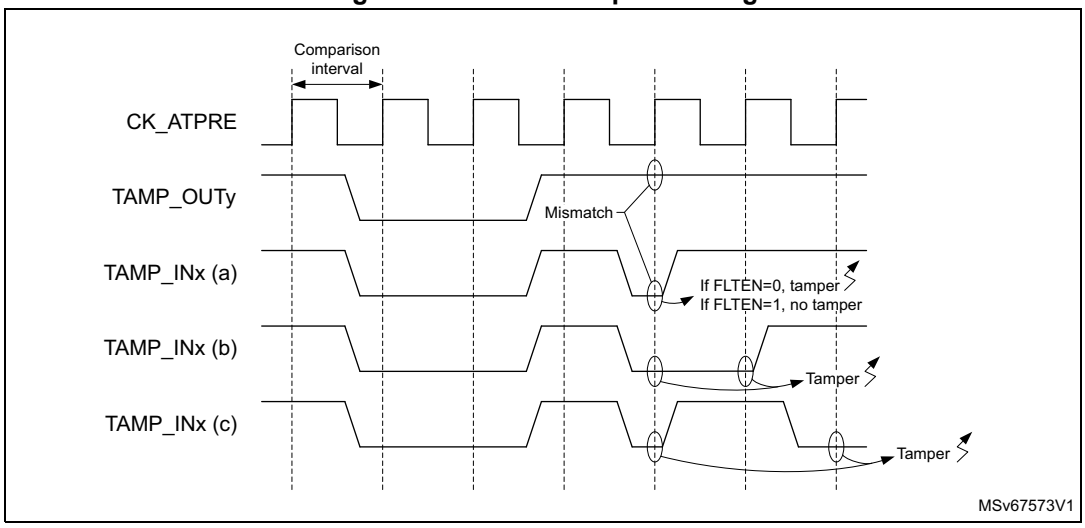
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

Figure 629. Active tamper filtering



As illustrated in *Figure 629*, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.

Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in *Table 511* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 511. Active tamper filtered pulse duration

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0x0	f_{RTCCLK}	0.030
0x1	$f_{RTCCLK}/2$	0.061
0x2	$f_{RTCCLK}/4$	0.122
0x3	$f_{RTCCLK}/8$	0.244
0x4	$f_{RTCCLK}/16$	0.488
0x5	$f_{RTCCLK}/32$	0.977
0x6	$f_{RTCCLK}/64$	1.953
0x7	$f_{RTCCLK}/128$	3.906
0xB	$f_{RTCCLK}/2048$	62.500(1)

1. This setting requires that (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.

Note: Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

Caution: Entering RTC initialization mode stops CK_ATPRE and CK_ATPER clocks when ATCKSEL[3] = 1. Therefore, TAMP_OUTy pin stops toggling until INIT mode exit.

Refer to *Calendar initialization and configuration*.

Refer also to *RTC alarm A subsecond register (RTC_ALRMASSR)*, *RTC alarm B subsecond register (RTC_ALRMBSSR)*, *RTC alarm A binary mode register (RTC_ALRABINR)* and *RTC alarm B binary mode register (RTC_ALRBBINR)* in case RTC binary mode is used in conjunction with ATCKSEL[3] = 1.

Caution: The active tamper detection is no more functional in case of calendar overflow when ATCKSEL[3] = 1. It is mandatory to enable the internal tamper 5 on calendar overflow to ensure tamper protection.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):

  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the TAMP APB clock.
- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

47.4 TAMP low-power modes

Table 512. Effect of low-power modes on TAMP

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

Table 513. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_IN[8:1]	Yes(1)	Yes(1)
TAMP_OUT[8:1]	Yes(2)	Yes(3)

1. Only PC13, PI8, PA0, PA1 and PA2 are functional in Standby and V_{BAT} modes.
2. Only PC13, PA1 and PI8 are functional in Standby mode.
3. Only PC13, PA1 and PI8 are functional in V_{BAT} mode.

47.5 TAMP interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration (TAMPSEC).

Table 514. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 512: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

47.6 TAMP registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

47.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7:*

TAMP privilege protection modes.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15E	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E
	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8E	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15E**: Internal tamper 15 enable
0: Internal tamper 15 disabled.
1: Internal tamper 15 enabled.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13E**: Internal tamper 13 enable
0: Internal tamper 13 disabled.
1: Internal tamper 13 enabled.

Bit 27 **ITAMP12E**: Internal tamper 12 enable
0: Internal tamper 12 disabled.
1: Internal tamper 12 enabled.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
0: Internal tamper 7 disabled.
1: Internal tamper 7 enabled

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
0: Internal tamper 2 disabled.
1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
0: Internal tamper 1 disabled.
1: Internal tamper 1 enabled.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8E**: Tamper detection on TAMP_IN8 enable(1)
0: Tamper detection on TAMP_IN8 is disabled.
1: Tamper detection on TAMP_IN8 is enabled.

Bit 6 **TAMP7E**: Tamper detection on TAMP_IN7 enable(1)
0: Tamper detection on TAMP_IN7 is disabled.
1: Tamper detection on TAMP_IN7 is enabled.

Bit 5 **TAMP6E**: Tamper detection on TAMP_IN6 enable(1)
0: Tamper detection on TAMP_IN6 is disabled.
1: Tamper detection on TAMP_IN6 is enabled.

Bit 4 **TAMP5E**: Tamper detection on TAMP_IN5 enable(1)
0: Tamper detection on TAMP_IN5 is disabled.
1: Tamper detection on TAMP_IN5 is enabled.

Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable(1)
0: Tamper detection on TAMP_IN4 is disabled.
1: Tamper detection on TAMP_IN4 is enabled.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

47.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7:*

TAMP privilege protection modes.

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP8 TRG	TAMP7 TRG	TAMP6 TRG	TAMP5 TRG	TAMP4 TRG	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
rw	rw	rw	rw	rw	rw	rw	rw	w	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8 POM	TAMP7 POM	TAMP6 POM	TAMP5 POM	TAMP4 POM	TAMP3 POM	TAMP2 POM	TAMP1 POM
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMP8TRG**: Active level for tamper 8 input (active mode disabled)

0: If TAMPFLT ? 00 tamper 8 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 8 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 8 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 8 input falling edge triggers a tamper detection event.

Bit 30 **TAMP7TRG**: Active level for tamper 7 input (active mode disabled)

0: If TAMPFLT ? 00 tamper 7 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 7 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 7 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 7 input falling edge triggers a tamper detection event.

Bit 29 **TAMP6TRG**: Active level for tamper 6 input (active mode disabled)

0: If TAMPFLT ? 00 tamper 6 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 6 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 6 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 6 input falling edge triggers a tamper detection event.

Bit 28 **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)

0: If TAMPFLT ? 00 tamper 5 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 5 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 5 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 5 input falling edge triggers a tamper detection event.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)

0: If TAMPFLT ? 00 tamper 4 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 4 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 4 input falling edge triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input

0: If TAMPFLT ? 00 tamper 3 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 3 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input

0: If TAMPFLT ? 00 tamper 2 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 2 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input

0: If TAMPFLT ? 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.

1: If TAMPFLT ? 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase

Writing '1' to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked

0: backup registers and device secrets(1) can be accessed if no tamper flag is set

1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask

0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.

1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask

0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.

1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.

The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8POM**: Tamper 8 potential mode

0: Tamper 8 event detection is in confirmed mode(1).

1: Tamper 8 event detection is in potential mode(2).

Bit 6 **TAMP7POM**: Tamper 7 potential mode

0: Tamper 7 event detection is in confirmed mode(1).

1: Tamper 7 event detection is in potential mode(2).

Bit 5 **TAMP6POM**: Tamper 6 potential mode

0: Tamper 6 event detection is in confirmed mode(1).

1: Tamper 6 event detection is in potential mode(2).

Bit 4 **TAMP5POM**: Tamper 5 potential mode
0: Tamper 5 event detection is in confirmed mode(1).
1: Tamper 5 event detection is in potential mode(2).

Bit 3 **TAMP4POM**: Tamper 4 potential mode
0: Tamper 4 event detection is in confirmed mode(1).
1: Tamper 4 event detection is in potential mode(2).

Bit 2 **TAMP3POM**: Tamper 3 potential mode
0: Tamper 3 event detection is in confirmed mode(1).
1: Tamper 3 event detection is in potential mode(2).

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode(1).
1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode(1).
1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 507: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 507: TAMP interconnection*.

47.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ITAMP 15POM	Res.	ITAMP 13POM	ITAMP 12POM	ITAMP 11POM	Res.	ITAMP9 POM	ITAMP8 POM	ITAMP7 POM	ITAMP6 POM	ITAMP5 POM	ITAMP4 POM	ITAMP3 POM	ITAMP2 POM	ITAMP1 POM
	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ITAMP15POM**: Internal tamper 15 potential mode
0: Internal tamper 15 event detection is in confirmed mode(1).
1: Internal tamper 15 event detection is in potential mode(2).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ITAMP13POM**: Internal tamper 13 potential mode
0: Internal tamper 13 event detection is in confirmed mode(1).
1: Internal tamper 13 event detection is in potential mode(2).

Bit 11 **ITAMP12POM**: Internal tamper 12 potential mode
0: Internal tamper 12 event detection is in confirmed mode(1).
1: Internal tamper 12 event detection is in potential mode(2).

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode(1).
1: Internal tamper 8 event detection is in potential mode(2).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode(1).
1: Internal tamper 7 event detection is in potential mode(2).

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 **ITAMP2POM**: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode(1).
1: Internal tamper 2 event detection is in potential mode(2).

Bit 0 **ITAMP1POM**: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode(1).
1: Internal tamper 1 event detection is in potential mode(2).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 507: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 507: TAMP interconnection*.

47.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP*

secure protection modes.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

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

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable
This bit determines if each of the TAMPx pins are precharged before each sample.
0: Precharge TAMP_INx pins before sampling (enable internal pull-up)
1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration
These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.
0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count
These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.
0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency
Determines the frequency at which each of the TAMP_INx inputs are sampled.
0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

47.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x0007 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	ATCKSEL[3:0]			
rw	rw				rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL4[1:0]		ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		TAMP8 AM	TAMP7 AM	TAMP6 AM	TAMP5 AM	TAMP4 AM	TAMP3 AM	TAMP2 AM	TAMP1 AM
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLTEN**: Active tamper filter enable

0: Active tamper filtering disable

1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing

0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi

1: Each active tamper input TAMP_INi is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period

The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 510: Minimum ATPER value*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection

These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE.

0000: RTCCLK is selected
0001: RTCCLK/2 is selected
0010: RTCCLK/4 is selected
0011: RTCCLK/8 is selected
0100: RTCCLK/16 is selected
0101: RTCCLK/32 is selected
0110: RTCCLK/64 is selected
0111: RTCCLK/128 is selected
1011: RTCCLK/2048 is selected when (PREDIV_A+1) = 128 and (PREDIV_S+1) is a multiple of 16.
Others: Reserved

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK_ATPRE cycles after all the active tampers are disable.

Bits 15:14 **ATOSEL4[1:0]**: Active tamper shared output 4 selection

00: TAMPOUTSEL4 = TAMP_OUT1
01: TAMPOUTSEL4 = TAMP_OUT2
10: TAMPOUTSEL4 = TAMP_OUT3
11: TAMPOUTSEL4 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 13:12 **ATOSEL3[1:0]**: Active tamper shared output 3 selection

00: TAMPOUTSEL3 = TAMP_OUT1
01: TAMPOUTSEL3 = TAMP_OUT2
10: TAMPOUTSEL3 = TAMP_OUT3
11: TAMPOUTSEL3 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection

00: TAMPOUTSEL2 = TAMP_OUT1
01: TAMPOUTSEL2 = TAMP_OUT2
10: TAMPOUTSEL2 = TAMP_OUT3
11: TAMPOUTSEL2 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection

00: TAMPOUTSEL1 = TAMP_OUT1
01: TAMPOUTSEL1 = TAMP_OUT2
10: TAMPOUTSEL1 = TAMP_OUT3
11: TAMPOUTSEL1 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bit 7 **TAMP8AM**: Tamper 8 active mode

0: Tamper 8 detection mode is passive.
1: Tamper 8 detection mode is active.

Bit 6 **TAMP7AM**: Tamper 7 active mode
0: Tamper 7 detection mode is passive.
1: Tamper 7 detection mode is active.

Bit 5 **TAMP6AM**: Tamper 6 active mode
0: Tamper 6 detection mode is passive.
1: Tamper 6 detection mode is active.

Bit 4 **TAMP5AM**: Tamper 5 active mode
0: Tamper 5 detection mode is passive.
1: Tamper 5 detection mode is active.

Bit 3 **TAMP4AM**: Tamper 4 active mode
0: Tamper 4 detection mode is passive.
1: Tamper 4 detection mode is active.

Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

47.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

47.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

This field can only be read when the APB is in secure mode.

47.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ATOSEL8[2:0]			ATOSEL7[2:0]			ATOSEL6[2:0]			ATOSEL5[2:0]			ATOSEL4[2:0]			ATOSEL3[2]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL3[1:0]		ATOSEL2[2:0]			ATOSEL1[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:29 **ATOSEL8[2:0]**: Active tamper shared output 8 selection
000: TAMPOUTSEL8 = TAMP_OUT1
001: TAMPOUTSEL8 = TAMP_OUT2
010: TAMPOUTSEL8 = TAMP_OUT3
011: TAMPOUTSEL8 = TAMP_OUT4
100: TAMPOUTSEL8 = TAMP_OUT5
101: TAMPOUTSEL8 = TAMP_OUT6
110: TAMPOUTSEL8 = TAMP_OUT7
111: TAMPOUTSEL8 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 28:26 **ATOSEL7[2:0]**: Active tamper shared output 7 selection
000: TAMPOUTSEL7 = TAMP_OUT1
001: TAMPOUTSEL7 = TAMP_OUT2
010: TAMPOUTSEL7 = TAMP_OUT3
011: TAMPOUTSEL7 = TAMP_OUT4
100: TAMPOUTSEL7 = TAMP_OUT5
101: TAMPOUTSEL7 = TAMP_OUT6
110: TAMPOUTSEL7 = TAMP_OUT7
111: TAMPOUTSEL7 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 25:23 **ATOSEL6[2:0]**: Active tamper shared output 6 selection
000: TAMPOUTSEL6 = TAMP_OUT1
001: TAMPOUTSEL6 = TAMP_OUT2
010: TAMPOUTSEL6 = TAMP_OUT3
011: TAMPOUTSEL6 = TAMP_OUT4
100: TAMPOUTSEL6 = TAMP_OUT5
101: TAMPOUTSEL6 = TAMP_OUT6
110: TAMPOUTSEL6 = TAMP_OUT7
111: TAMPOUTSEL6 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 22:20 **ATOSEL5[2:0]**: Active tamper shared output 5 selection
000: TAMPOUTSEL5 = TAMP_OUT1
001: TAMPOUTSEL5 = TAMP_OUT2
010: TAMPOUTSEL5 = TAMP_OUT3
011: TAMPOUTSEL5 = TAMP_OUT4
100: TAMPOUTSEL5 = TAMP_OUT5
101: TAMPOUTSEL5 = TAMP_OUT6
110: TAMPOUTSEL5 = TAMP_OUT7
111: TAMPOUTSEL5 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection

000: TAMPOUTSEL4 = TAMP_OUT1
001: TAMPOUTSEL4 = TAMP_OUT2
010: TAMPOUTSEL4 = TAMP_OUT3
011: TAMPOUTSEL4 = TAMP_OUT4
100: TAMPOUTSEL4 = TAMP_OUT5
101: TAMPOUTSEL4 = TAMP_OUT6
110: TAMPOUTSEL4 = TAMP_OUT7
111: TAMPOUTSEL4 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 18:17 are the mirror of ATOSEL4[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection

000: TAMPOUTSEL3 = TAMP_OUT1
001: TAMPOUTSEL3 = TAMP_OUT2
010: TAMPOUTSEL3 = TAMP_OUT3
011: TAMPOUTSEL3 = TAMP_OUT4
100: TAMPOUTSEL3 = TAMP_OUT5
101: TAMPOUTSEL3 = TAMP_OUT6
110: TAMPOUTSEL3 = TAMP_OUT7
111: TAMPOUTSEL3 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 15:14 are the mirror of ATOSEL3[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 13:11 **ATOSEL2[2:0]**: Active tamper shared output 2 selection
000: TAMPOUTSEL2 = TAMP_OUT1
001: TAMPOUTSEL2 = TAMP_OUT2
010: TAMPOUTSEL2 = TAMP_OUT3
011: TAMPOUTSEL2 = TAMP_OUT4
100: TAMPOUTSEL2 = TAMP_OUT5
101: TAMPOUTSEL2 = TAMP_OUT6
110: TAMPOUTSEL2 = TAMP_OUT7
111: TAMPOUTSEL2 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 12:11 are the mirror of ATOSEL2[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 10:8 **ATOSEL1[2:0]**: Active tamper shared output 1 selection
000: TAMPOUTSEL1 = TAMP_OUT1
001: TAMPOUTSEL1 = TAMP_OUT2
010: TAMPOUTSEL1 = TAMP_OUT3
011: TAMPOUTSEL1 = TAMP_OUT4
100: TAMPOUTSEL1 = TAMP_OUT5
101: TAMPOUTSEL1 = TAMP_OUT6
110: TAMPOUTSEL1 = TAMP_OUT7
111: TAMPOUTSEL1 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 9:8 are the mirror of ATOSEL1[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

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

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

47.6.9 TAMP secure configuration register (TAMP_SECCFGR)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0] and BHKLOCK can be written with nonsecure APB access, and TAMPSEC, CNT1SEC cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 47.3.7: TAMP privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
rw	rs							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPSEC**: Tamper protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written when the APB access is secure or nonsecure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 47.3.5: TAMP secure protection modes for details on the read protection.

Bit 30 **BHKLOCK**: Boot hardware key lock

This bit can be read and can only be written to 1 by software. It is cleared by hardware together with the backup registers following a tamper detection event or when the product state is opened.

0: The Backup registers from TAMP_BKP0R to TAMP_BKP7R can be accessed according to the Protection zone they belong to.

1: The backup registers from TAMP_BKP0R to TAMP_BKP7R cannot be accessed neither in read nor in write (they are read as 0 and write ignore).

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWSEC[7:0]**: Backup registers write protection offset

BKPWSEC value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPzR (z = BKPWSEC-1, with BKPWSEC > BKPRWSEC):

- if TZEN=1, these backup registers can be written only with secure access.
- They can be read with secure or nonsecure access.

If BKPWSEC = 0 or if BKPWSEC ? BKPRWSEC: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPWSEC if BKPWSEC ? BKPRWSEC, else t = BKPRWSEC).

- They can be read or written with secure or nonsecure access.

If BKPWSEC = 32: there is no protection zone 3.

Refer to *Figure 626: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 2 can be read and written with nonsecure access.

Note: If BKPWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or nonsecure.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup registers read/write protection offset
BKPRWSEC value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC-1, with BKPRWSEC ? 1).
– if TZEN=1, these backup registers can be read and written only with secure access.
If BKPRWSEC = 0: there is no protection zone 1.

Refer to *Figure 626: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 1 can be read and written with nonsecure access.
Note: If BKPRWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

47.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the TAMP_SECCFGR configuration (refer to *Section 47.3.5: TAMP secure protection modes*).

Address offset: 0x24

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1 PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding backup registers)
0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
1: Tamper configuration and interrupt can be written only with privileged access.
Note: Refer to Section 47.3.7: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection
0: Backup registers zone 2 can be written with privileged or unprivileged access.
1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection
0: Backup registers zone 1 can be read and written with privileged or unprivileged access.
1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or non-privileged.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bits 14:0 Reserved, must be kept at reset value.

47.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15IE	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE
	rw		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8IE	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15IE**: Internal tamper 15 interrupt enable
0: Internal tamper 15 interrupt disabled.
1: Internal tamper 15 interrupt enabled.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13IE**: Internal tamper 13 interrupt enable
0: Internal tamper 13 interrupt disabled.
1: Internal tamper 13 interrupt enabled.

Bit 27 **ITAMP12IE**: Internal tamper 12 interrupt enable
0: Internal tamper 12 interrupt disabled.
1: Internal tamper 12 interrupt enabled.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8IE**: Tamper 8 interrupt enable
0: Tamper 8 interrupt disabled.
1: Tamper 8 interrupt enabled.

Bit 6 **TAMP7IE**: Tamper 7interrupt enable
0: Tamper 7 interrupt disabled.
1: Tamper 7interrupt enabled.

Bit 5 **TAMP6IE**: Tamper 6 interrupt enable
0: Tamper 6 interrupt disabled.
1: Tamper 6 interrupt enabled.

Bit 4 **TAMP5IE**: Tamper 5 interrupt enable
0: Tamper 5 interrupt disabled.
1: Tamper 5 interrupt enabled.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

47.6.12 TAMP status register (TAMP_SR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15F	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F
	rw		r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8F	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
								r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15F**: Internal tamper 15 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 15.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13F**: Internal tamper 13 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 13.

Bit 27 **ITAMP12F**: Internal tamper 12 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 12.

Bit 26 **ITAMP11F**: Internal tamper 11 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 **ITAMP8F**: Internal tamper 8 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 **ITAMP7F**: Internal tamper 7 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8F**: TAMP8 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP8 input

Bit 6 **TAMP7F**: TAMP7 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP7 input.

Bit 5 **TAMP6F**: TAMP6 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP6 input.

Bit 4 **TAMP5F**: TAMP5 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3 **TAMP4F**: TAMP4 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

47.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15MF	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
	r		r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
								r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15MF**: internal tamper 15 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 15 nonsecure interrupt is raised.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13MF**: internal tamper 13 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 13 nonsecure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 12 nonsecure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 11 nonsecure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 9 nonsecure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 8 nonsecure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 7 nonsecure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 6 nonsecure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 5 nonsecure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 4 nonsecure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 3 nonsecure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 2 nonsecure interrupt is raised.

Bit 16 **ITAMP1MF**:Internal tamper 1 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 1 nonsecure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8MF**: TAMP8 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 8 nonsecure interrupt is raised.

Bit 6 **TAMP7MF**: TAMP7 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 7 nonsecure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 6 nonsecure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 5 nonsecure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 4 nonsecure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 3 nonsecure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 2 nonsecure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 1 nonsecure interrupt is raised.

47.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	ITAMP15MF	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
	r		r	r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
								r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15MF**: internal tamper 15 secure interrupt masked flag
This flag is set by hardware when the internal tamper 15 secure interrupt is raised.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **ITAMP13MF**: internal tamper 13 secure interrupt masked flag
This flag is set by hardware when the internal tamper 13 secure interrupt is raised.

Bit 27 **ITAMP12MF**: internal tamper 12 secure interrupt masked flag
This flag is set by hardware when the internal tamper 12 secure interrupt is raised.

Bit 26 **ITAMP11MF**: internal tamper 11 secure interrupt masked flag
This flag is set by hardware when the internal tamper 11 secure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 secure interrupt masked flag
This flag is set by hardware when the internal tamper 9 secure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 secure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 secure interrupt masked flag
This flag is set by hardware when the internal tamper 7 secure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 secure interrupt masked flag
This flag is set by hardware when the internal tamper 6 secure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 secure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 secure interrupt masked flag
This flag is set by hardware when the internal tamper 4 secure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 secure interrupt is raised.

Bit 16 **ITAMP1MF**: Internal tamper 1 secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 secure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8MF**: TAMP8 secure interrupt masked flag
This flag is set by hardware when the tamper 8 secure interrupt is raised.

Bit 6 **TAMP7MF**: TAMP7 secure interrupt masked flag
This flag is set by hardware when the tamper 7 secure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 secure interrupt masked flag
This flag is set by hardware when the tamper 6 secure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 secure interrupt masked flag
This flag is set by hardware when the tamper 5 secure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 secure interrupt masked flag
This flag is set by hardware when the tamper 4 secure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 secure interrupt masked flag
This flag is set by hardware when the tamper 3 secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag
This flag is set by hardware when the tamper 1 secure interrupt is raised.

47.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CITAMP15F	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F
	w		w	w	w		w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP8F	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
								w	w	w	w	w	w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CITAMP15F**: Clear ITAMP15 detection flag
Writing 1 in this bit clears the ITAMP15F bit in the TAMP_SR register.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CITAMP13F**: Clear ITAMP13 detection flag
Writing 1 in this bit clears the ITAMP13F bit in the TAMP_SR register.

Bit 27 **CITAMP12F**: Clear ITAMP12 detection flag
Writing 1 in this bit clears the ITAMP12F bit in the TAMP_SR register.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 **CITAMP7F**: Clear ITAMP7 detection flag
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CTAMP8F**: Clear TAMP8 detection flag
Writing 1 in this bit clears the TAMP8F bit in the TAMP_SR register.

Bit 6 **CTAMP7F**: Clear TAMP7 detection flag
Writing 1 in this bit clears the TAMP7F bit in the TAMP_SR register.

Bit 5 **CTAMP6F**: Clear TAMP6 detection flag
Writing 1 in this bit clears the TAMP6F bit in the TAMP_SR register.

Bit 4 **CTAMP5F**: Clear TAMP5 detection flag
Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3 **CTAMP4F**: Clear TAMP4 detection flag
Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

47.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **COUNT[31:0]:**

This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

47.6.17 TAMP option register (TAMP_OR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	IN4_RMP	IN3_RMP	IN2_RMP	Res.	Res.	Res.	Res.	OUT5_RMP	OUT3_RMP[1:0]		Res.
					rw	rw	rw					rw	rw	rw	

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **IN4_RMP**: TAMP_IN4 mapping
0: TAMP_IN4 is on PA2
1: TAMP_IN4 is on PI11

Bit 9 **IN3_RMP**: TAMP_IN3 mapping
0: TAMP_IN3 is on PC1
1: TAMP_IN3 is on PE6

Bit 8 **IN2_RMP**: TAMP_IN2 mapping
0: TAMP_IN2 is on PA0
1: TAMP_IN2 is on PI8

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **OUT5_RMP**: TAMP_OUT5 mapping
0: TAMP_OUT5 is on PI11
1: TAMP_OUT5 is on PC1

Bits 2:1 **OUT3_RMP[1:0]**: TAMP_OUT3 mapping
00: TAMP_OUT3 is on PC13
01: TAMP_OUT3 is on PI8
10: TAMP_OUT3 is on PE3
11: TAMP_OUT3 is on PA2

Bit 0 Reserved, must be kept at reset value.

47.6.18 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RP CFG0
															rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **RPCFG0**: Configurable resource 0 protection(1)

0: Resource 0 is not included in the device secrets protected by TAMP peripheral

1: Resource 0 is included in the device secrets protected by TAMP peripheral

1. Refer to tamp_confirmed_rpcfg0 and tamp_potential_rpcfg0 signals in *Table 505: TAMP input/output pins* and *Table 507: TAMP interconnection*.

47.6.19 TAMP backup x register (TAMP_BKPxR)

This register can be protected against nonsecure access. Refer to *Section 47.3.5: TAMP secure protection modes*.

This register can be protected against non-privileged access. Refer to *Section 47.3.7: TAMP privilege protection modes*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:0 **BKP[31:0]:**

The application can write or read data to and from these registers.
In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the product state is opened.

47.6.20 TAMP register map

Table 515. TAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TAMP_CR1	Res.	ITAMP15E	Res.	ITAMP13E	ITAMP12E	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8E	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x04	TAMP_CR2	TAMP8TRG	TAMP7TRG	TAMP6TRG	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8POM	TAMP7POM	TAMP6POM	TAMP5POM	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value	0	0	0	0	0	0	0	0	0	0				0	0	0									0	0	0	0	0	0	0	0
0x08	TAMP_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP15POM	Res.	ITAMP13POM	ITAMP12POM	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	ITAMP4POM	ITAMP3POM	ITAMP2POM	ITAMP1POM
	Reset value																		0		0	0	0		0	0	0	0	0	0	0	0	0
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	ATCK SEL[3:0]				ATO SEL4 [1:0]		ATO SEL3 [1:0]		ATO SEL2 [1:0]		ATO SEL1 [1:0]		TAMP8AM	TAMP7AM	TAMP6AM	TAMP5AM	TAMP4AM	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0					0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	TAMP_ ATSEEDR	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TAMP_ATOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x1C	TAMP_ATCR2	ATO SEL8 [2:0]			ATO SEL7 [2:0]			ATO SEL6 [2:0]			ATO SEL5 [2:0]			ATO SEL4 [2:0]				ATO SEL3 [2:0]			ATO SEL2 [2:0]			ATO SEL1 [2:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x20	TAMP_SEC CFGR	TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]								CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
	Reset value	0	0							0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0
0x24	TAMP_ PRIVCFGR	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0															

Table 515. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TAMP_IER	Res.	ITAMP15IE	Res.	ITAMP13IE	ITAMP12IE	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8IE	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x30	TAMP_SR	Res.	ITAMP15F	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8F	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x34	TAMP_MISR	Res.	ITAMP15MF	Res.	ITAMP13MF	ITAMP12MF	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x38	TAMP_SMISR	Res.	ITAMP15F	Res.	ITAMP13F	ITAMP12F	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP8MF	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x3C	TAMP_SCR	Res.	CITAMP15F	Res.	CITAMP13F	CITAMP12F	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP8F	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value		0		0	0	0		0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x40	TAMP_ COUNTR	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TAMP_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN4_RMP	IN3_RMP	IN2_RMP	Res.	Res.	Res.	Res.	OUT5_RMP	OUT3_RMP		Res.
	Reset value																						0	0	0					0	0	0	
0x54	TAMP_ RPCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG0
	Reset value																																0
0x100 + 0x04*x, (x= 0 to 31)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

48 Inter-integrated circuit interface (I2C)

48.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

48.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to *Section 48.3: I2C implementation*.

48.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 516. I2C implementation

I2C features[1]	I2C1	I2C2	I2C3	I2C4[2]
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Wake-up from Stop mode only (no autonomous mode)	X	X	X	X
SMBus/PMBus	X	X	X	X

1. X = supported.
2. I2C4 is only available on STM32H56xx/57xx devices.

48.4 I2C functional description

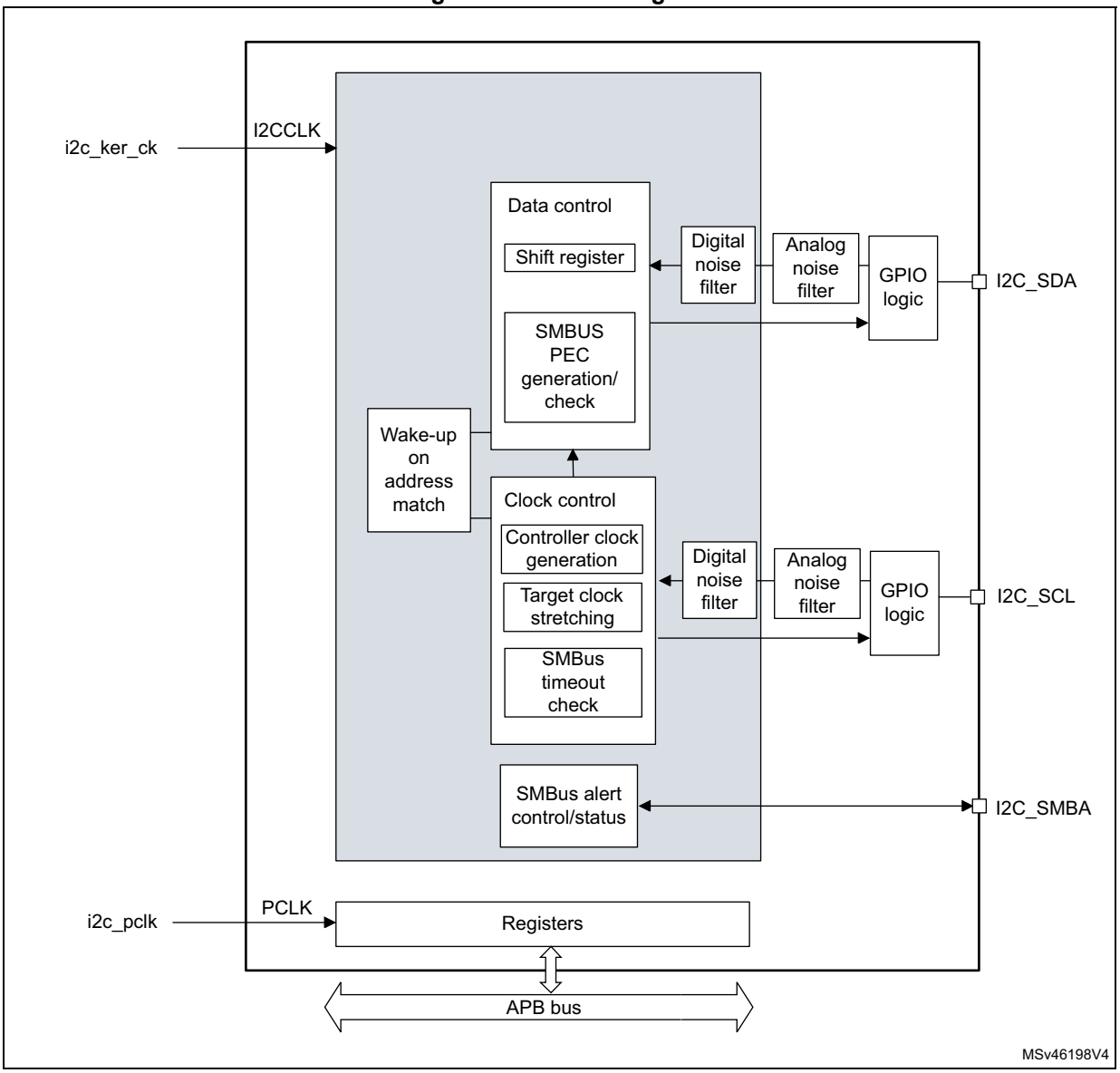
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

48.4.1 I2C block diagram

Figure 630. Block diagram



48.4.2 I2C pins and internal signals

Table 517. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 518. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 531* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

48.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

48.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

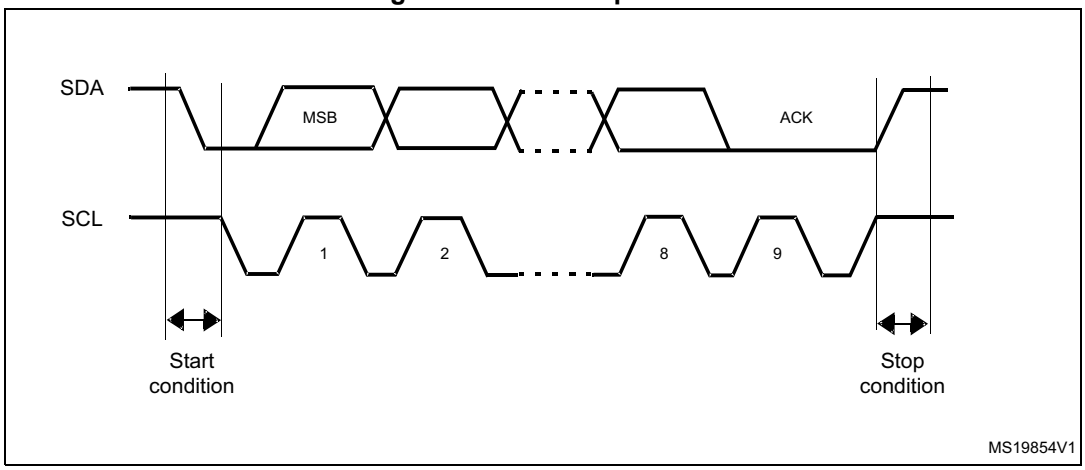
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 631. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

48.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 48.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 519. Comparison of analog and digital filters

Item	**Analog filter**	**Digital filter**
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

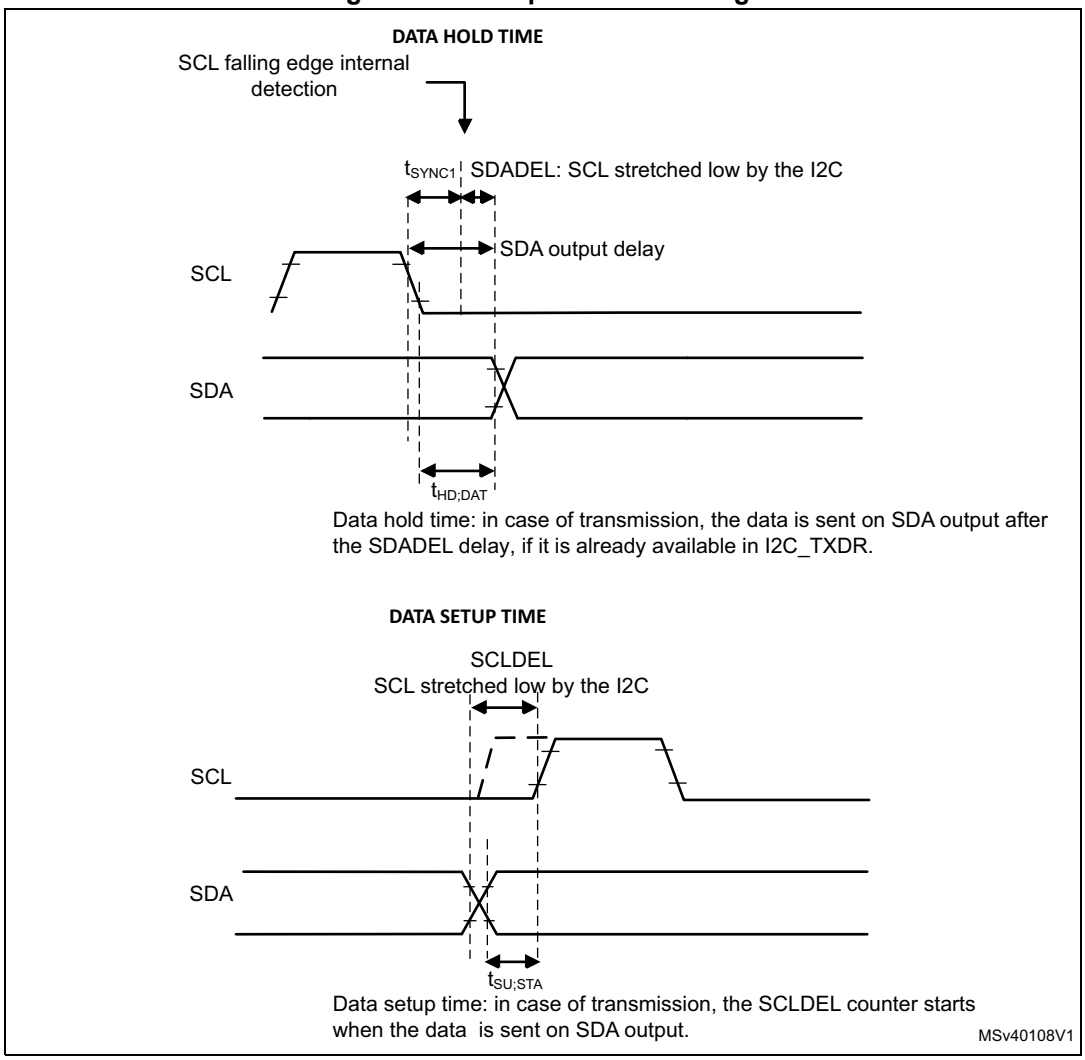
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 632. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

{$t_{f(max)}$ + $t_{HD;DAT(min)}$ - $t_{AF(min)}$ - [(DNF + 3) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}} ≤ SDADEL

SDADEL ≤ {$t_{HD;DAT\ (max)}$ - $t_{AF(max)}$ - [(DNF + 4) x t_{I2CCLK}]} / {(PRESC + 1) x t_{I2CCLK}}

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 520. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

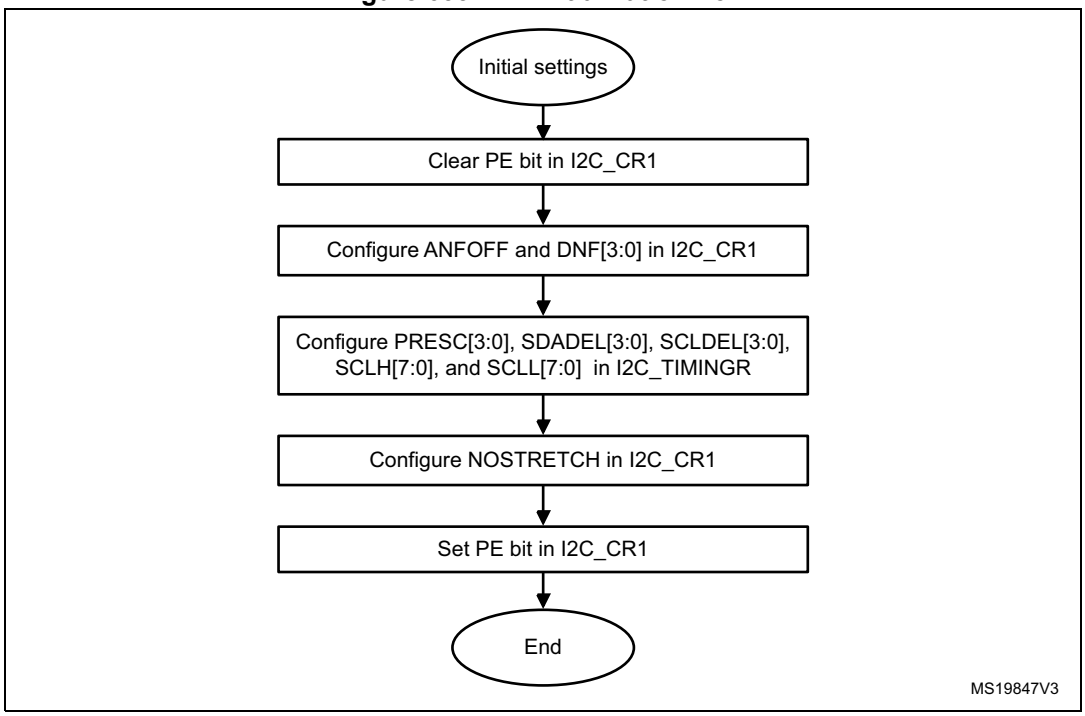
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 633. I2C initialization flow



48.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

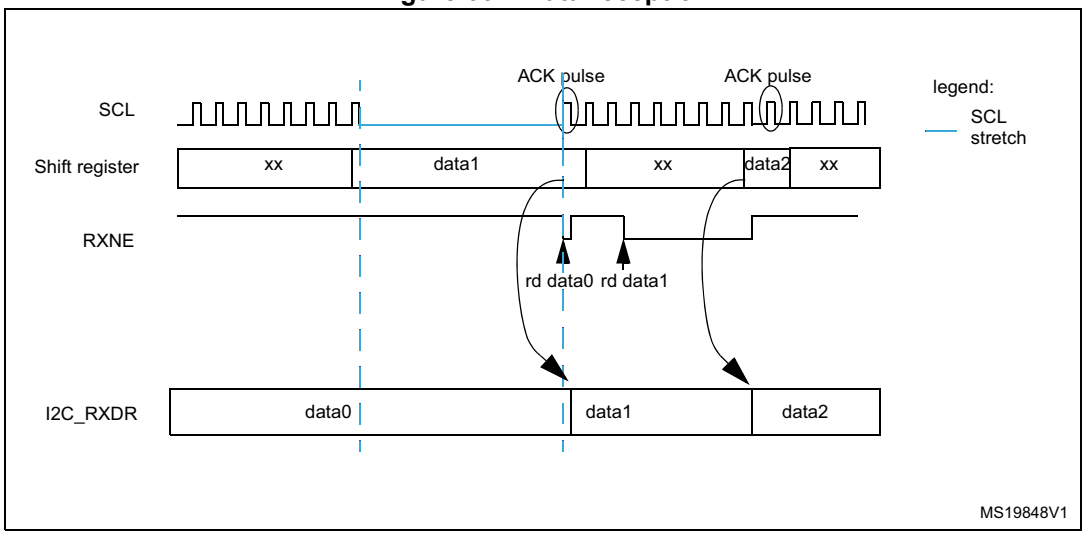
48.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

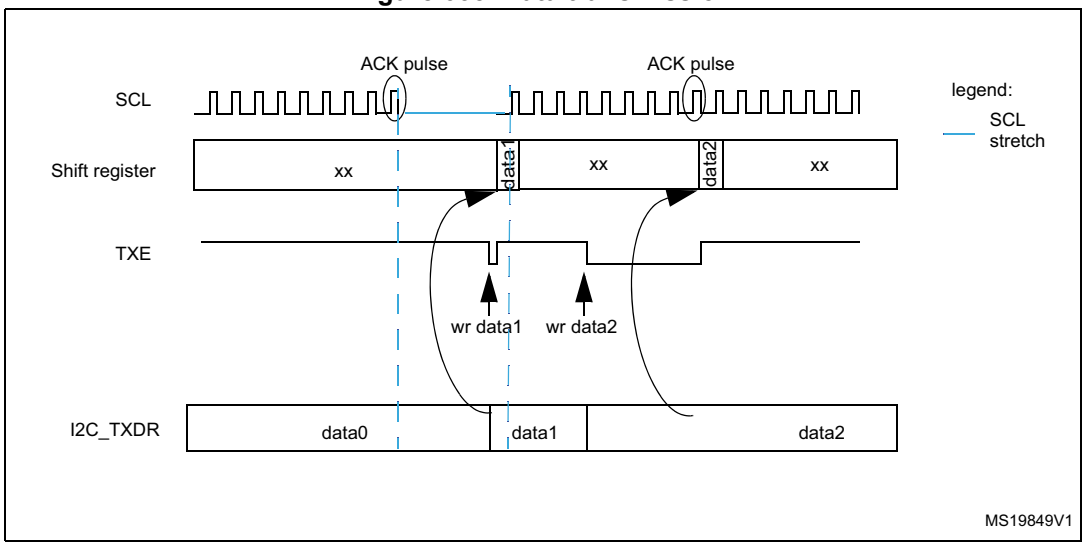
Figure 634. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 635. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 521. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

48.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

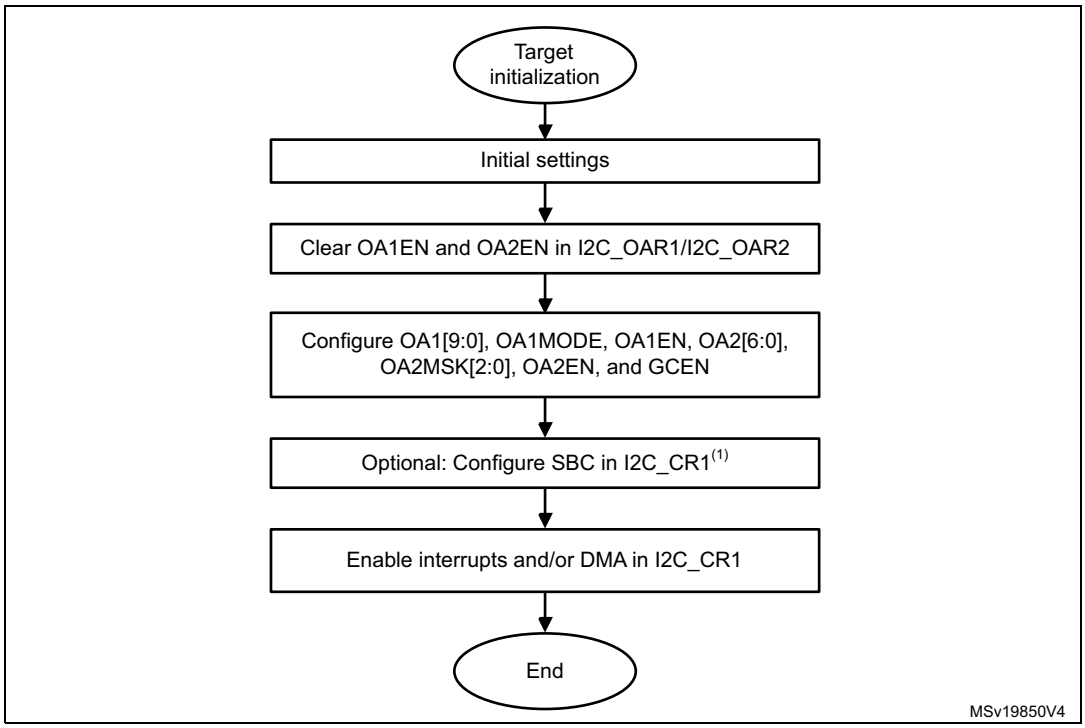
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 636. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 637. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

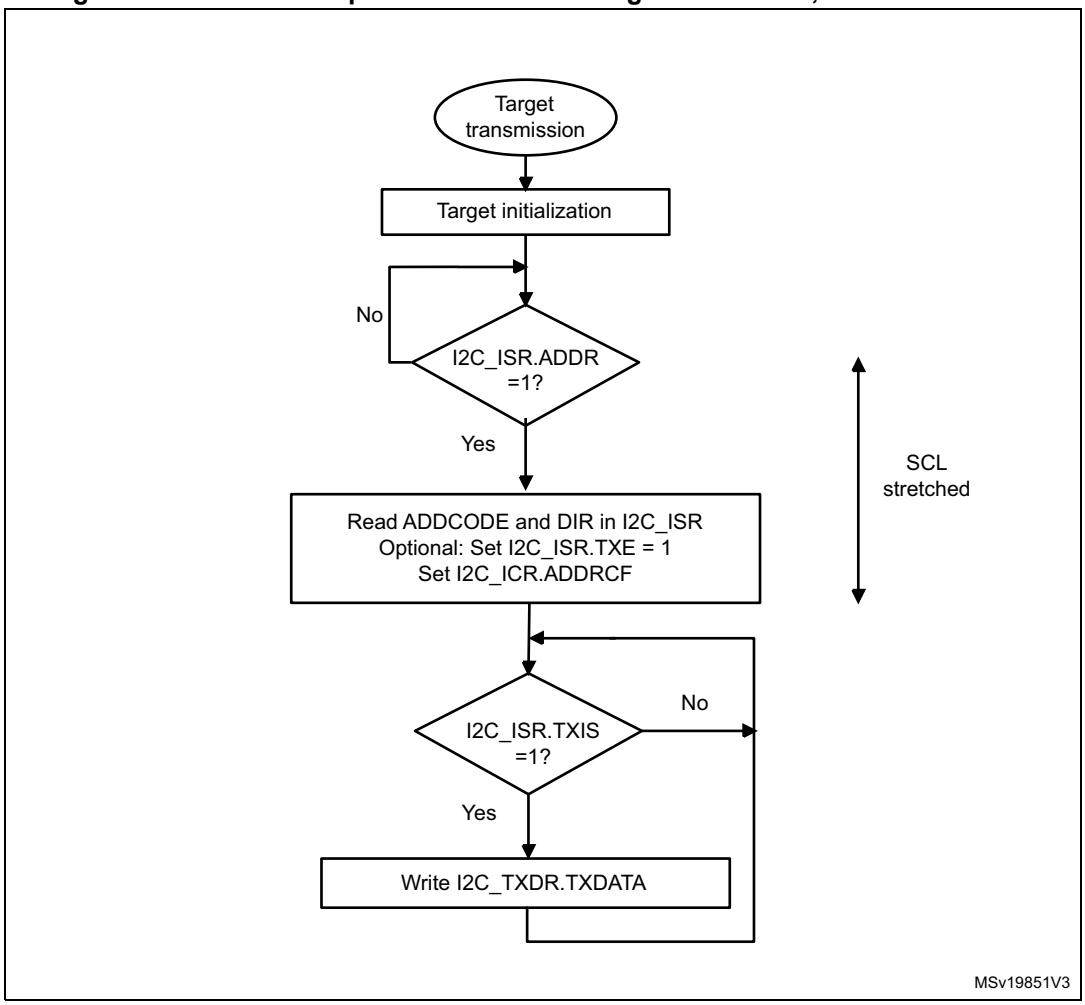


Figure 638. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

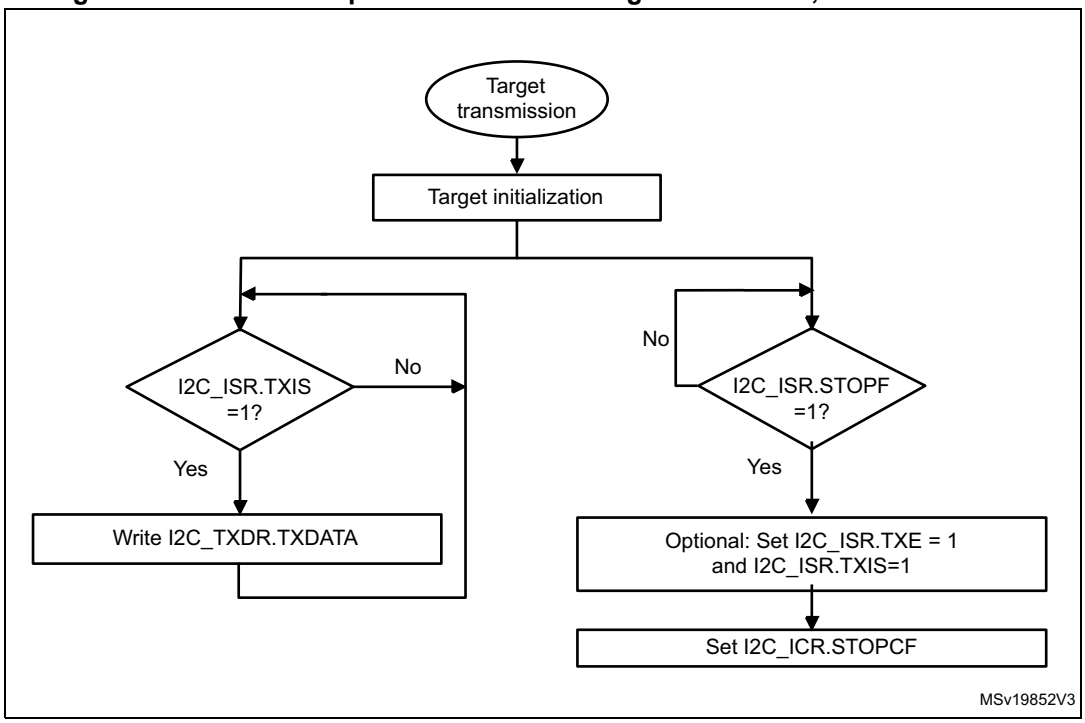
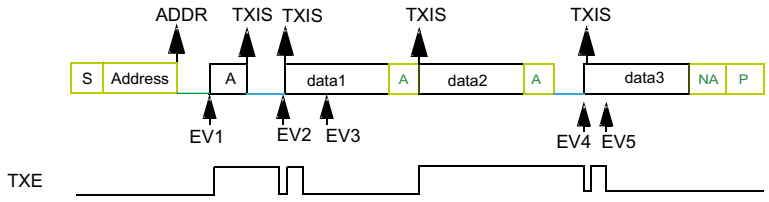


Figure 639. Transfer bus diagrams for I2C target transmitter (mandatory events only)

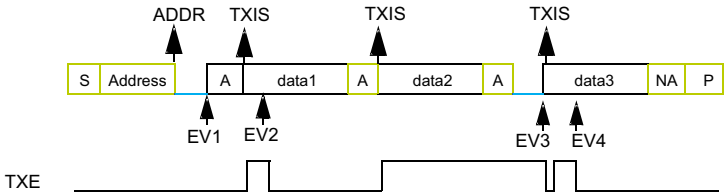
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

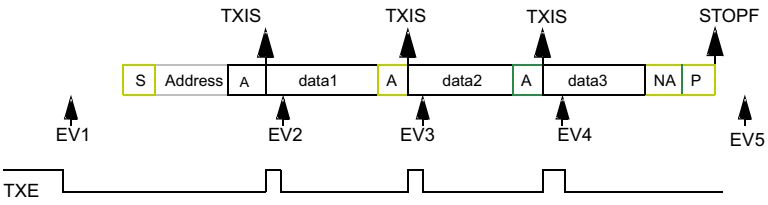
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 640. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

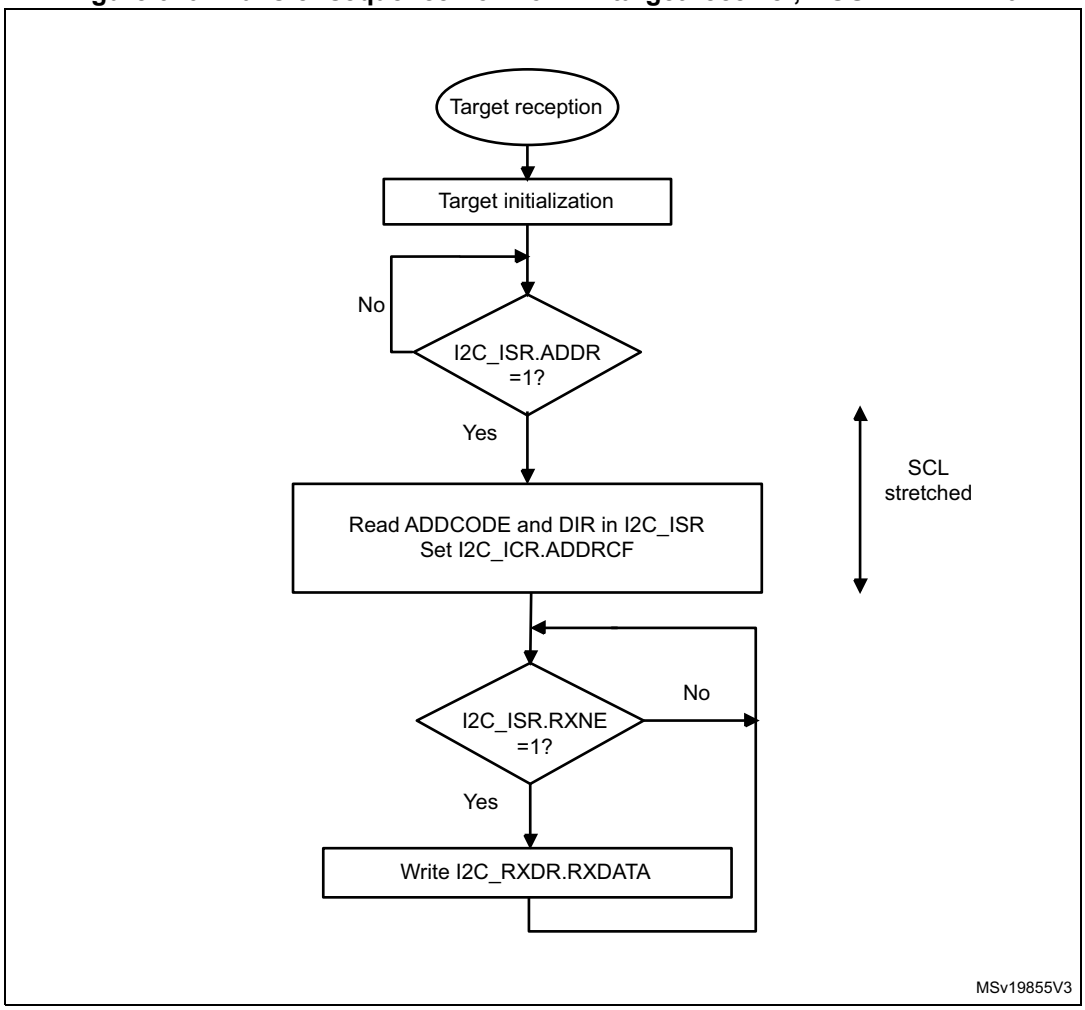


Figure 641. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

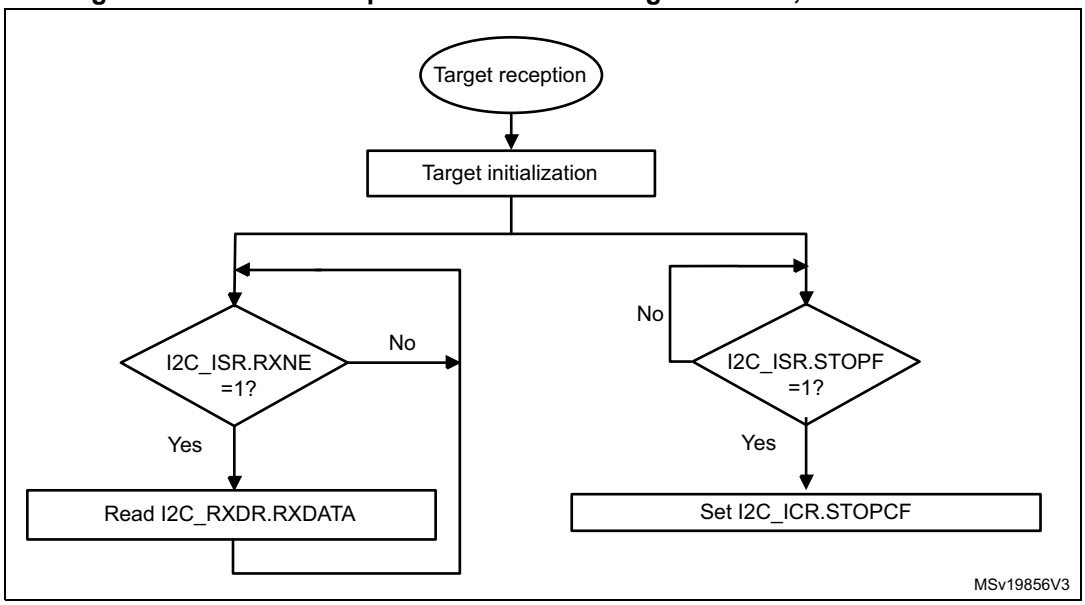
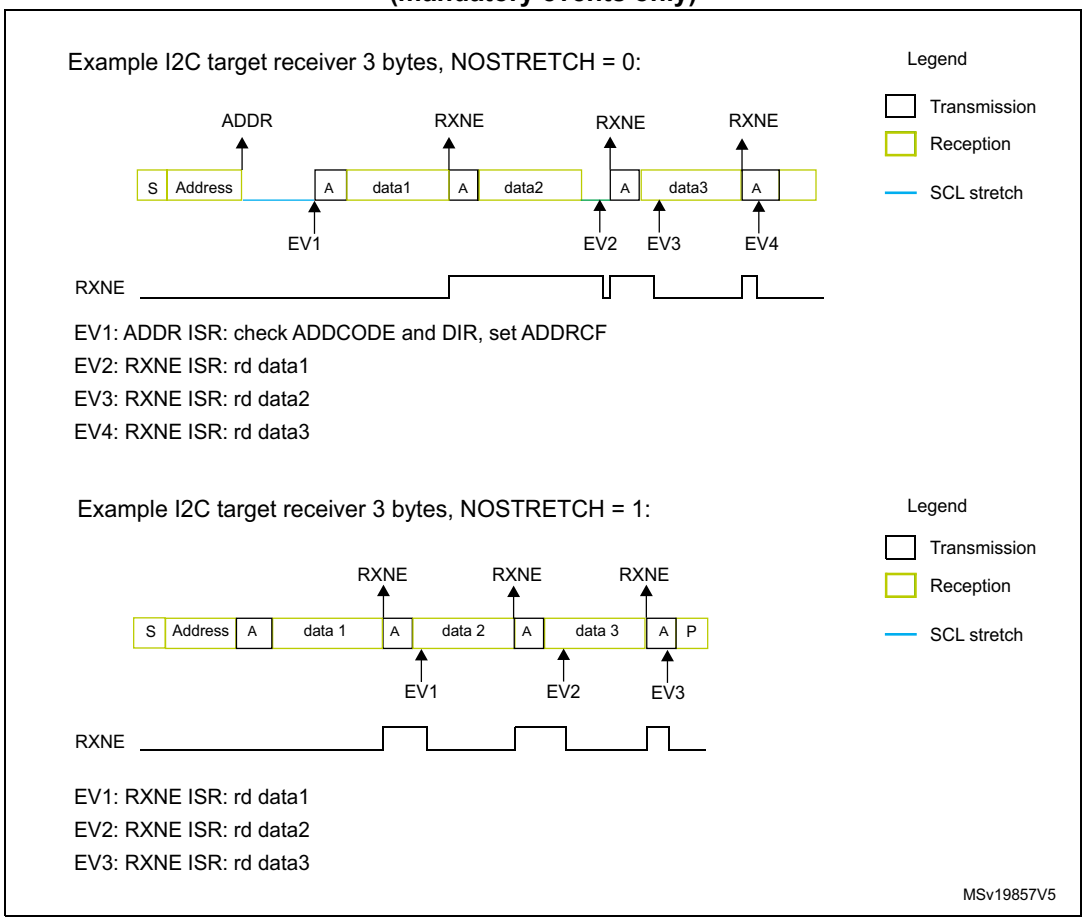


Figure 642. Transfer bus diagrams for I2C target receiver (mandatory events only)



48.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

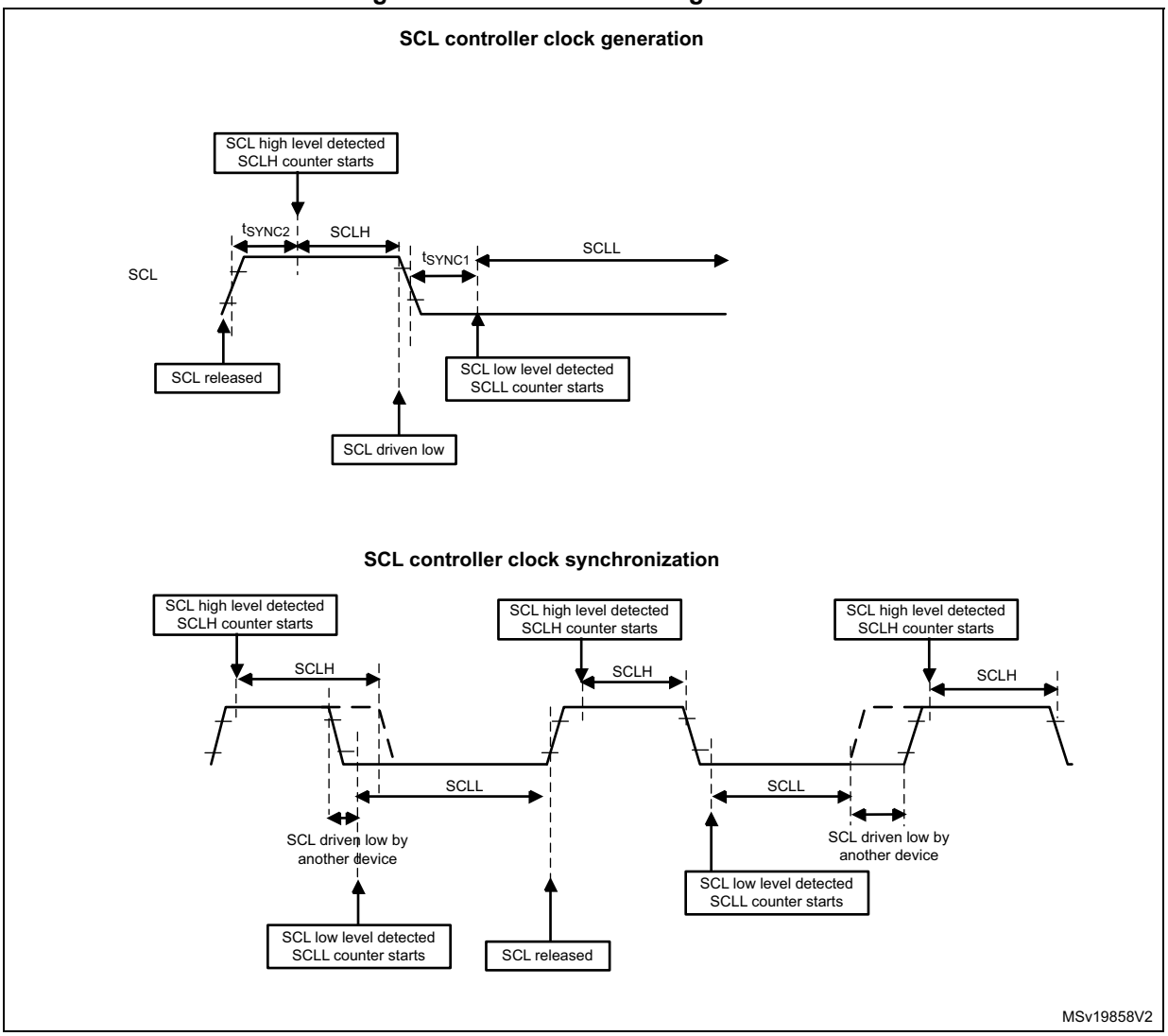
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 643. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 522. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 48.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

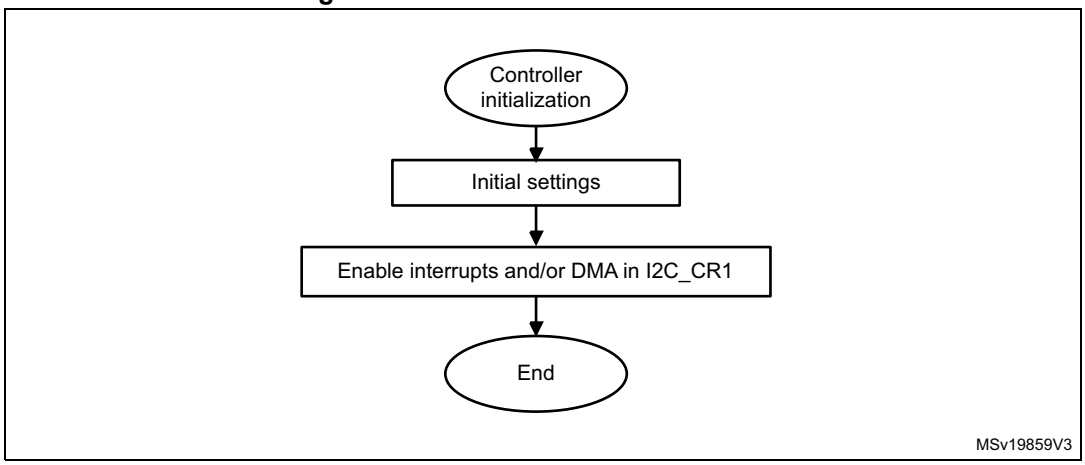
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 644. Controller initialization flow

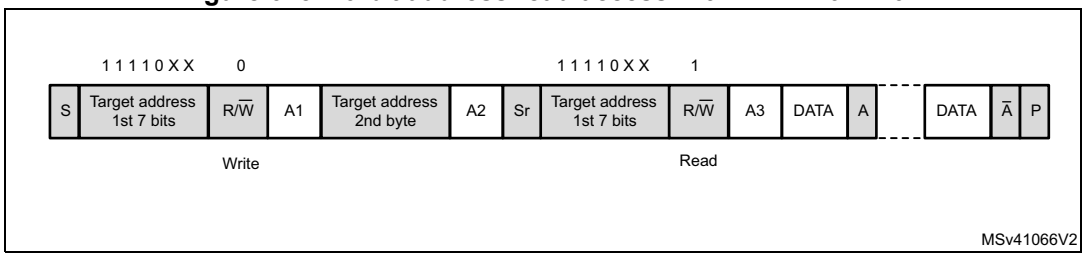


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

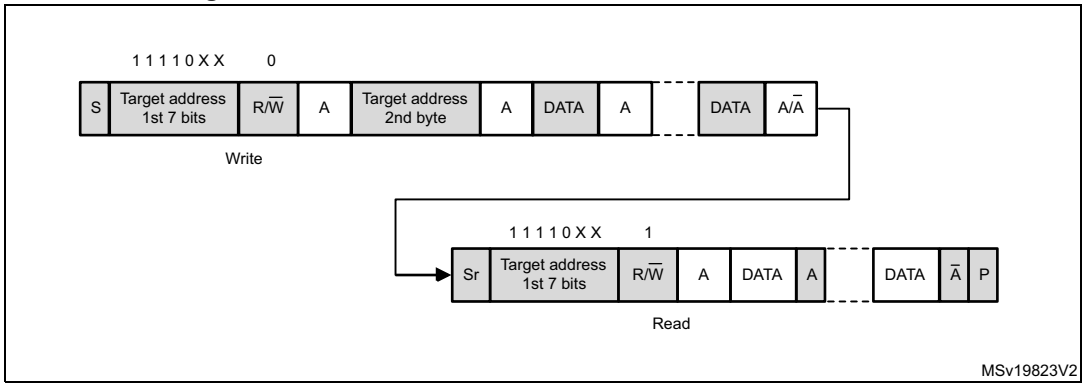
Figure 645. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 646. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 647. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

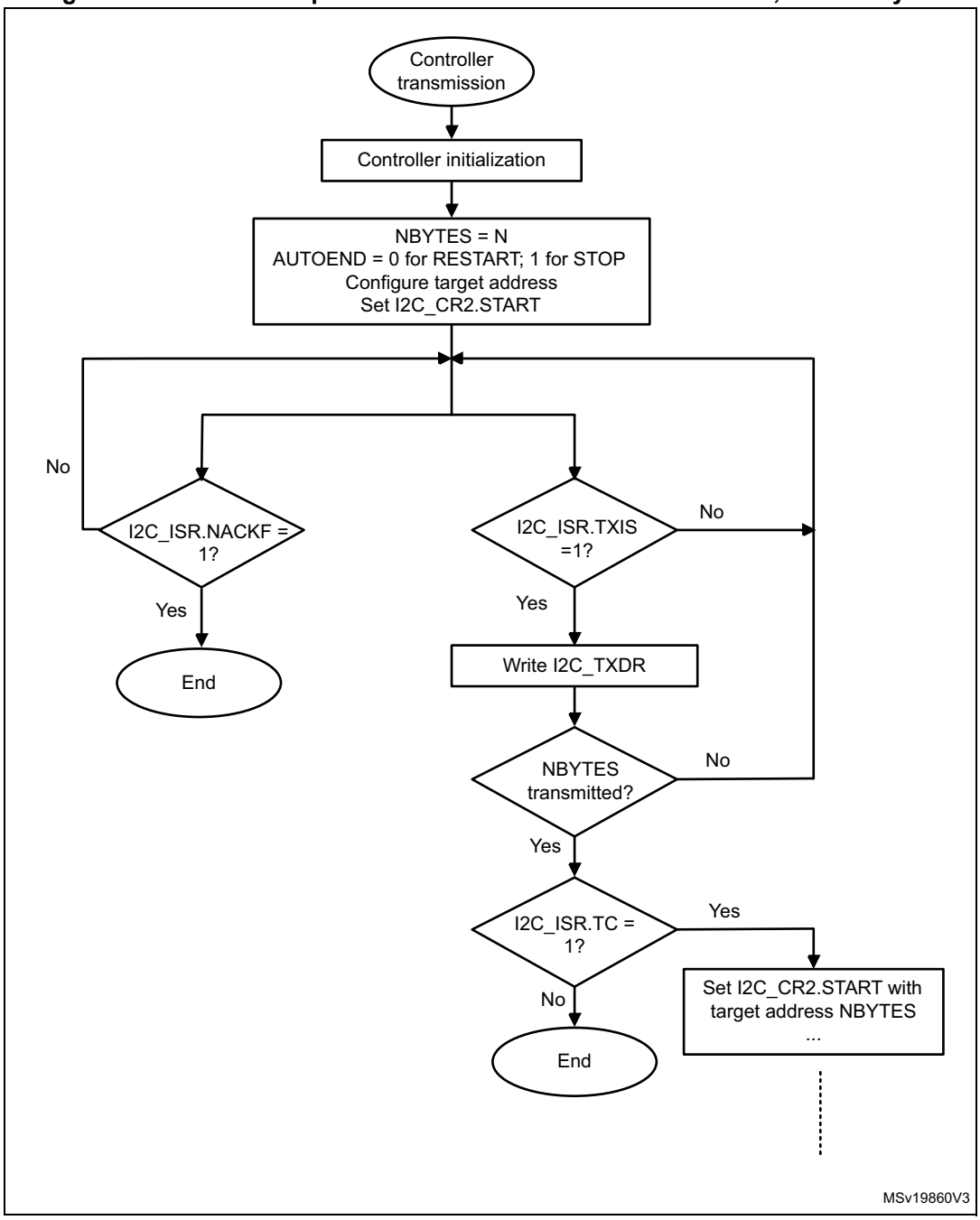


Figure 648. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

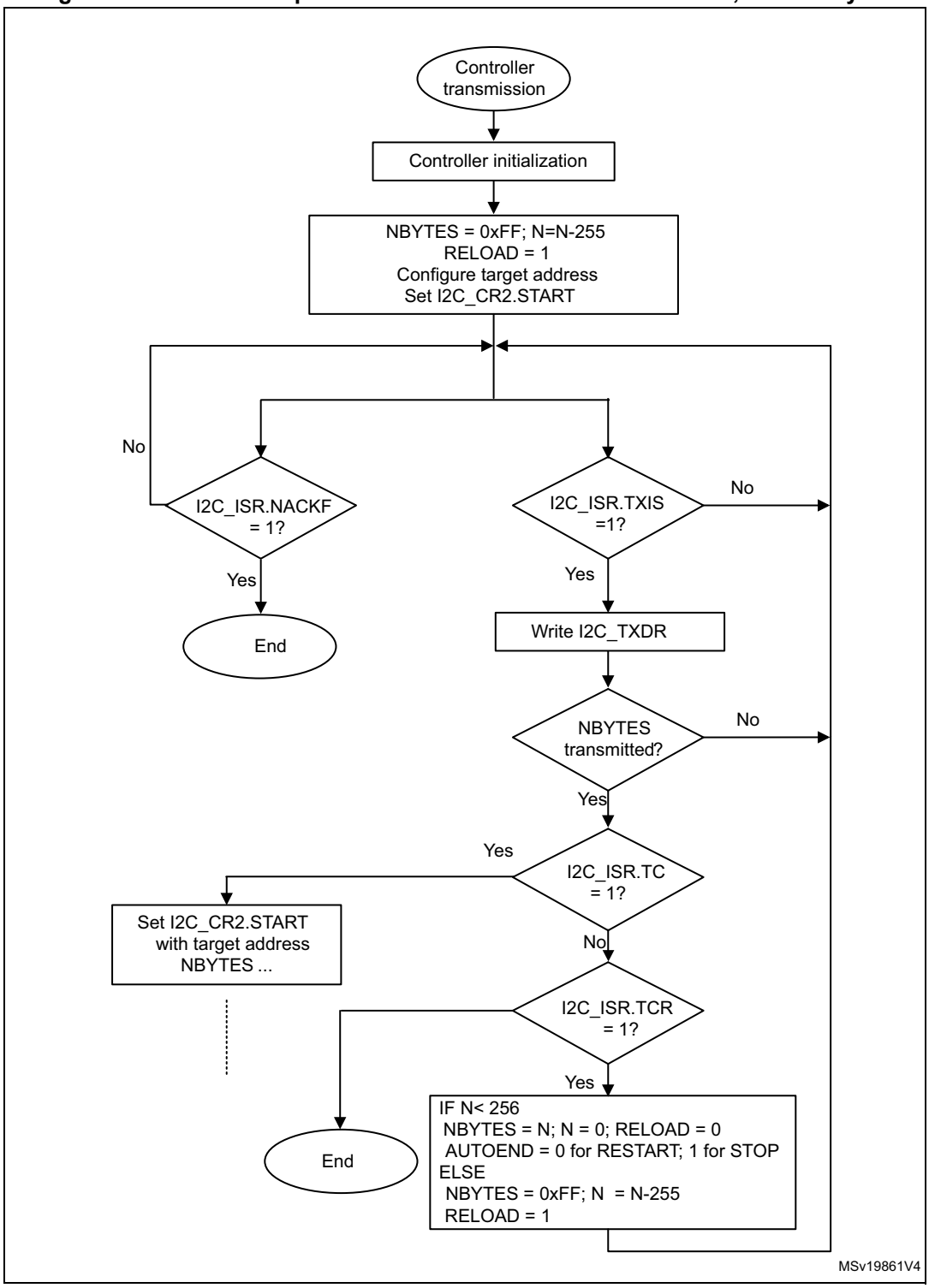
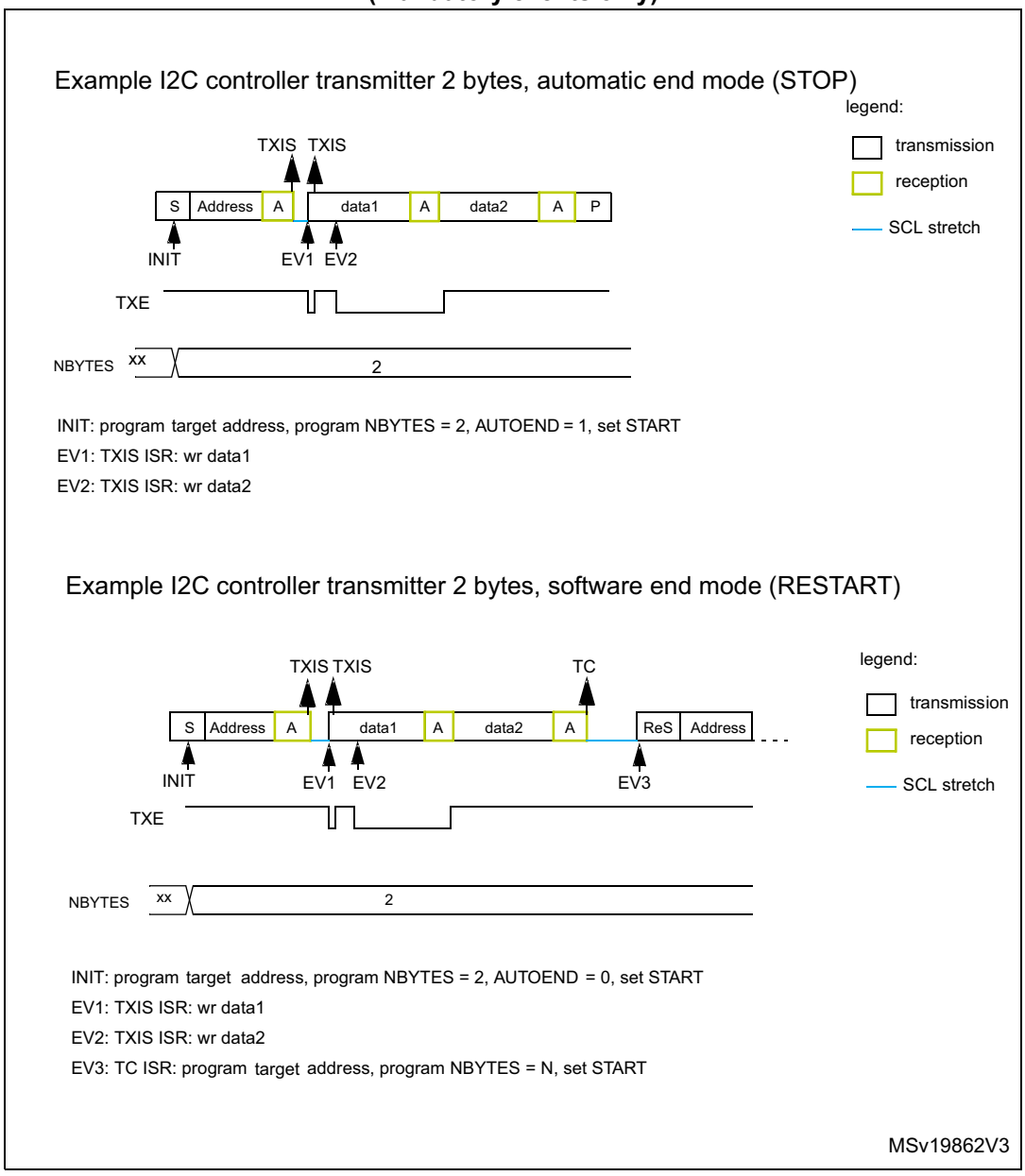


Figure 649. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 650. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

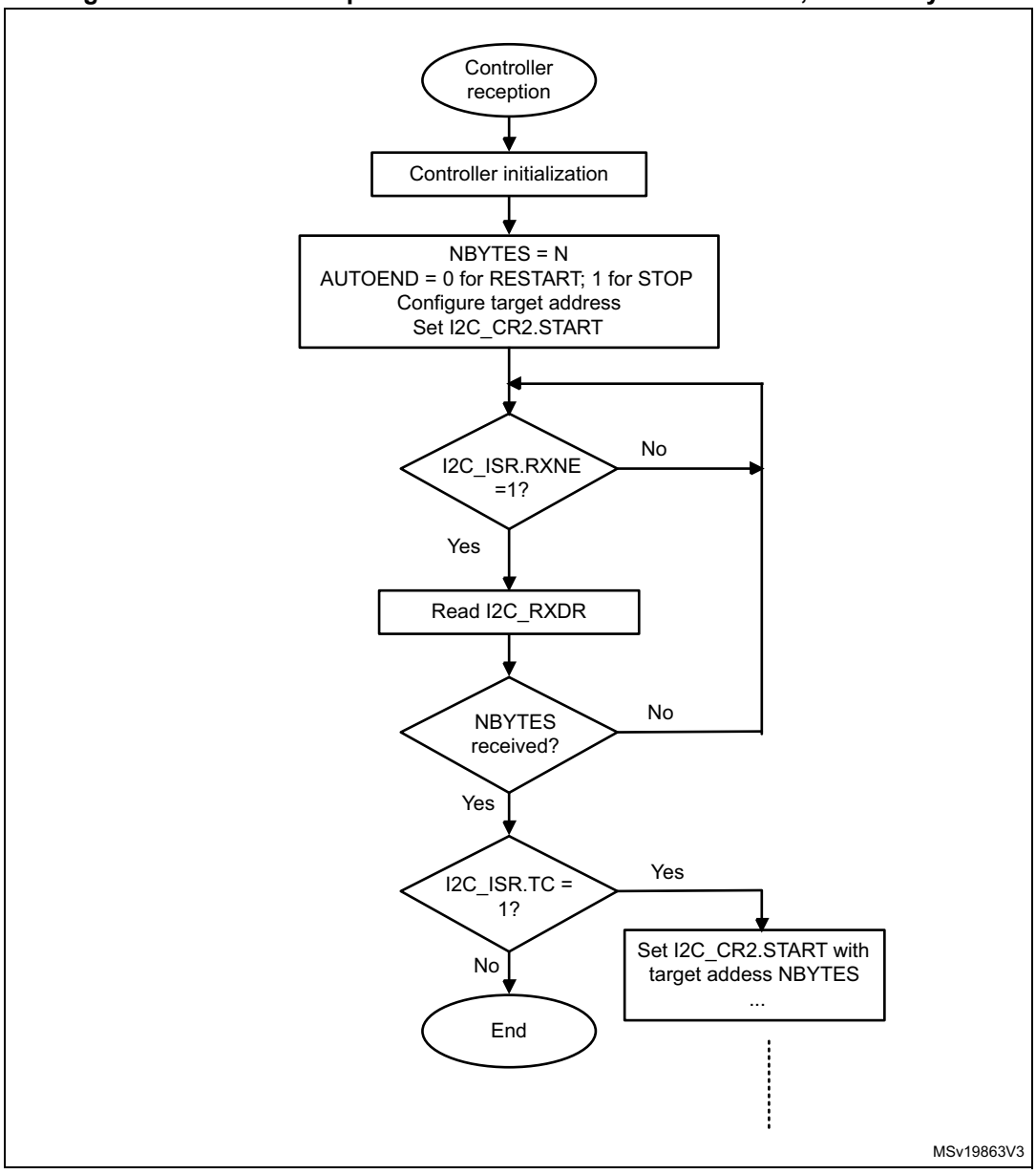


Figure 651. Transfer sequence flow for I2C controller receiver, N > 255 bytes

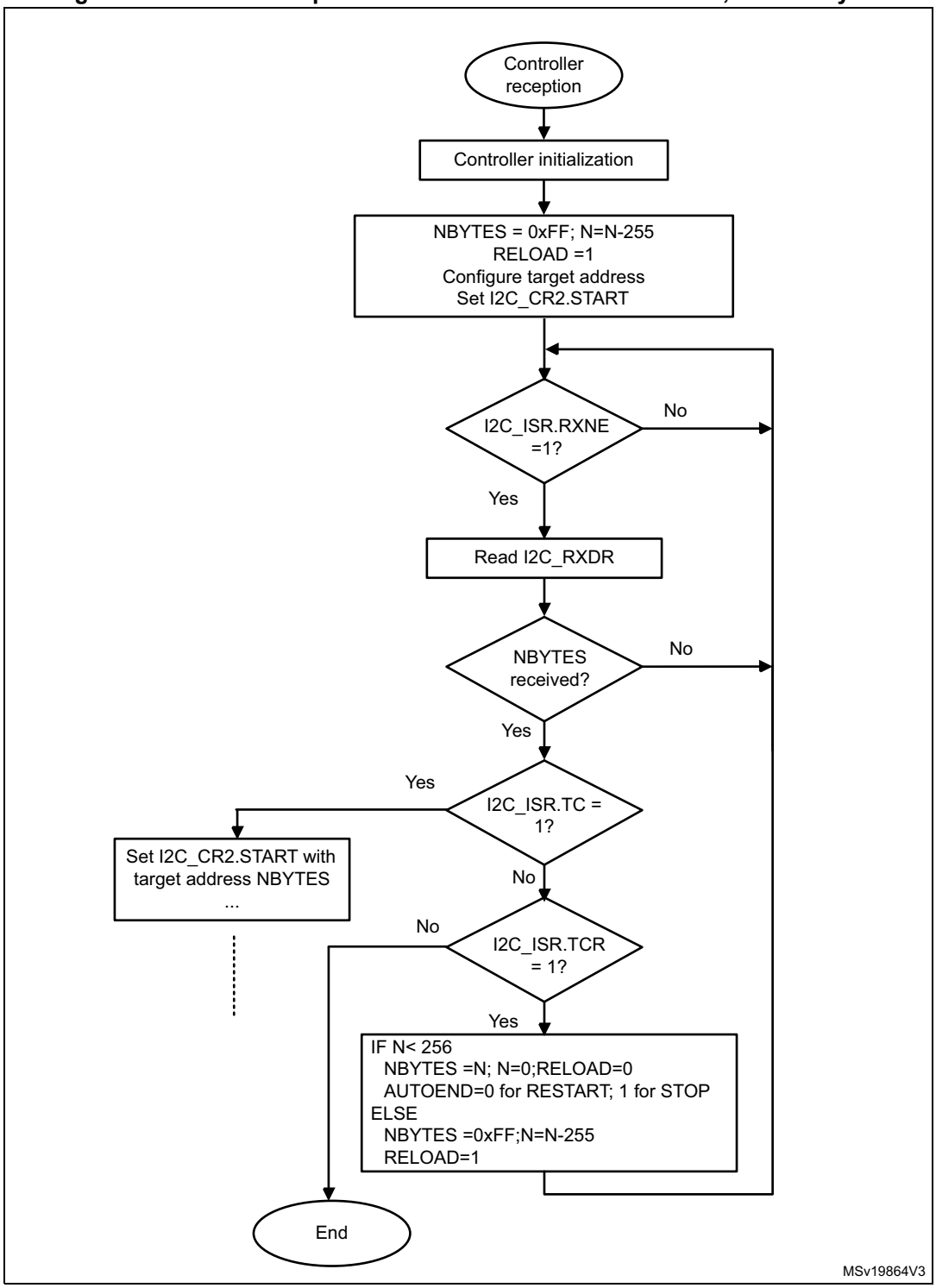
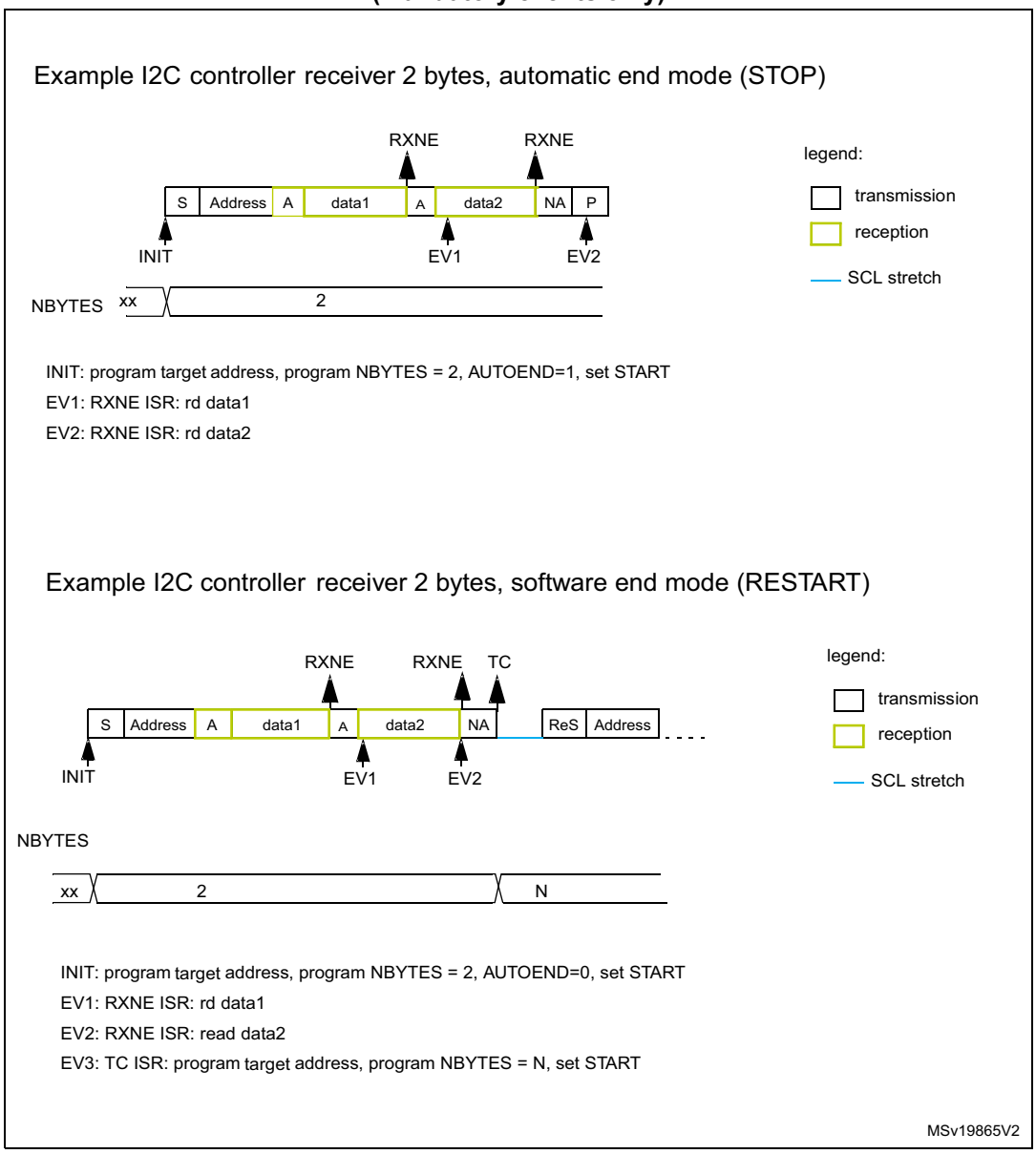


Figure 652. Transfer bus diagrams for I2C controller receiver (mandatory events only)



48.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 523. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 524. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

48.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

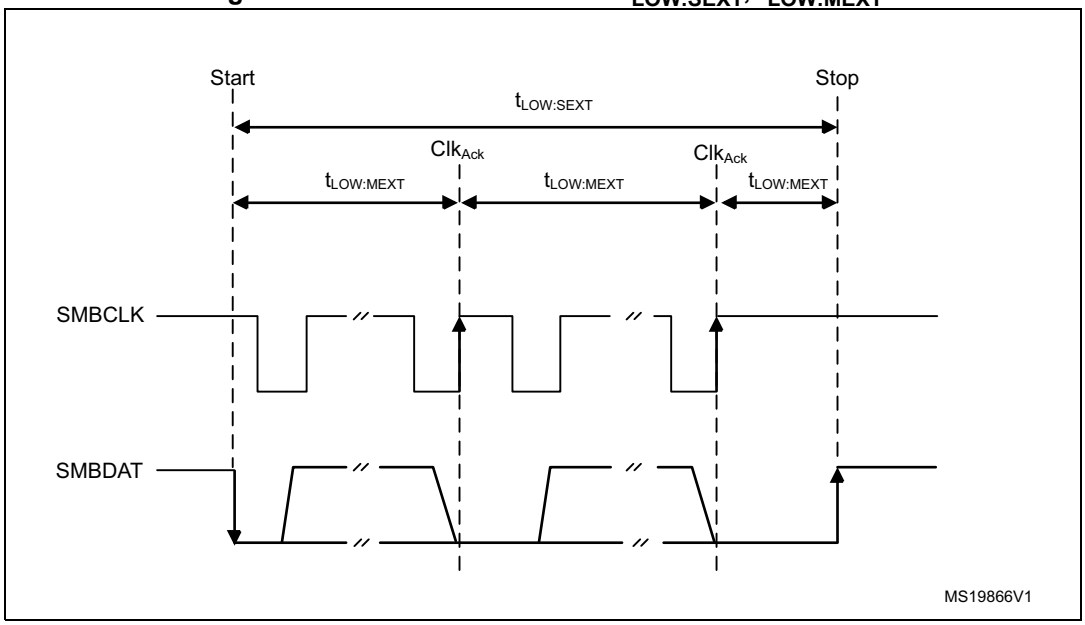
Table 525. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 653. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 48.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

48.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 526. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 527*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 528*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 529*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

48.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 527. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

Table 528. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

Table 529. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

48.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 48.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 654. Transfer sequence flow for SMBus target transmitter N bytes + PEC

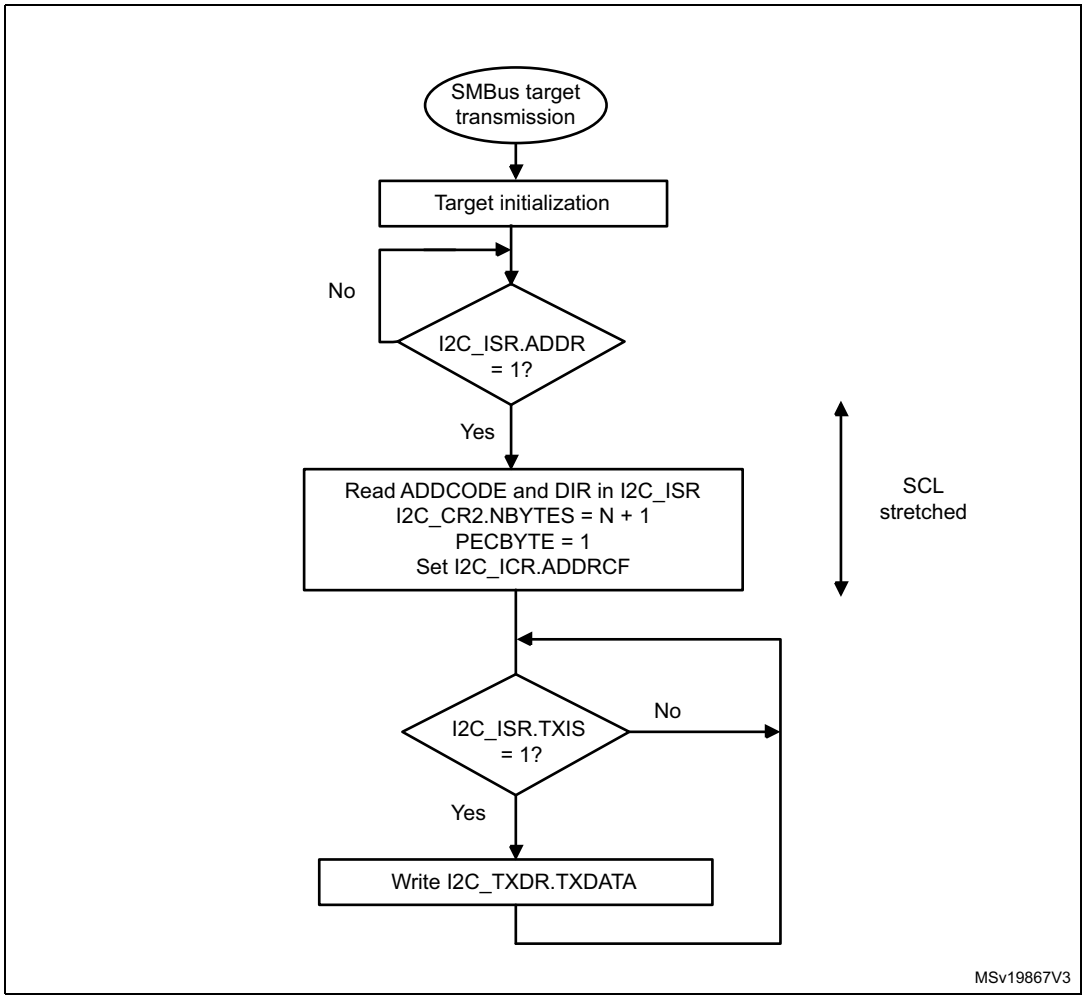
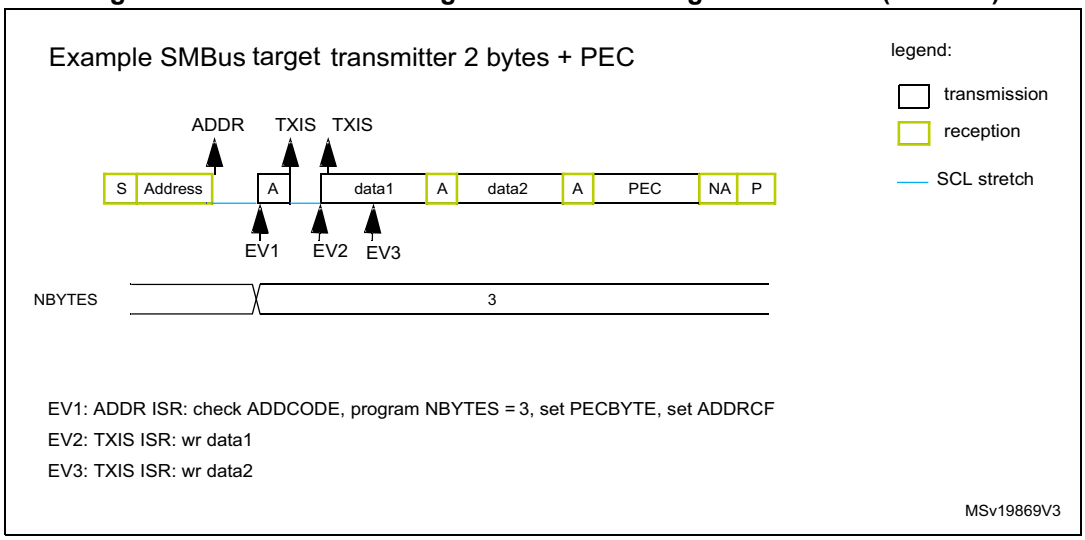


Figure 655. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 656. Transfer sequence flow for SMBus target receiver N bytes + PEC

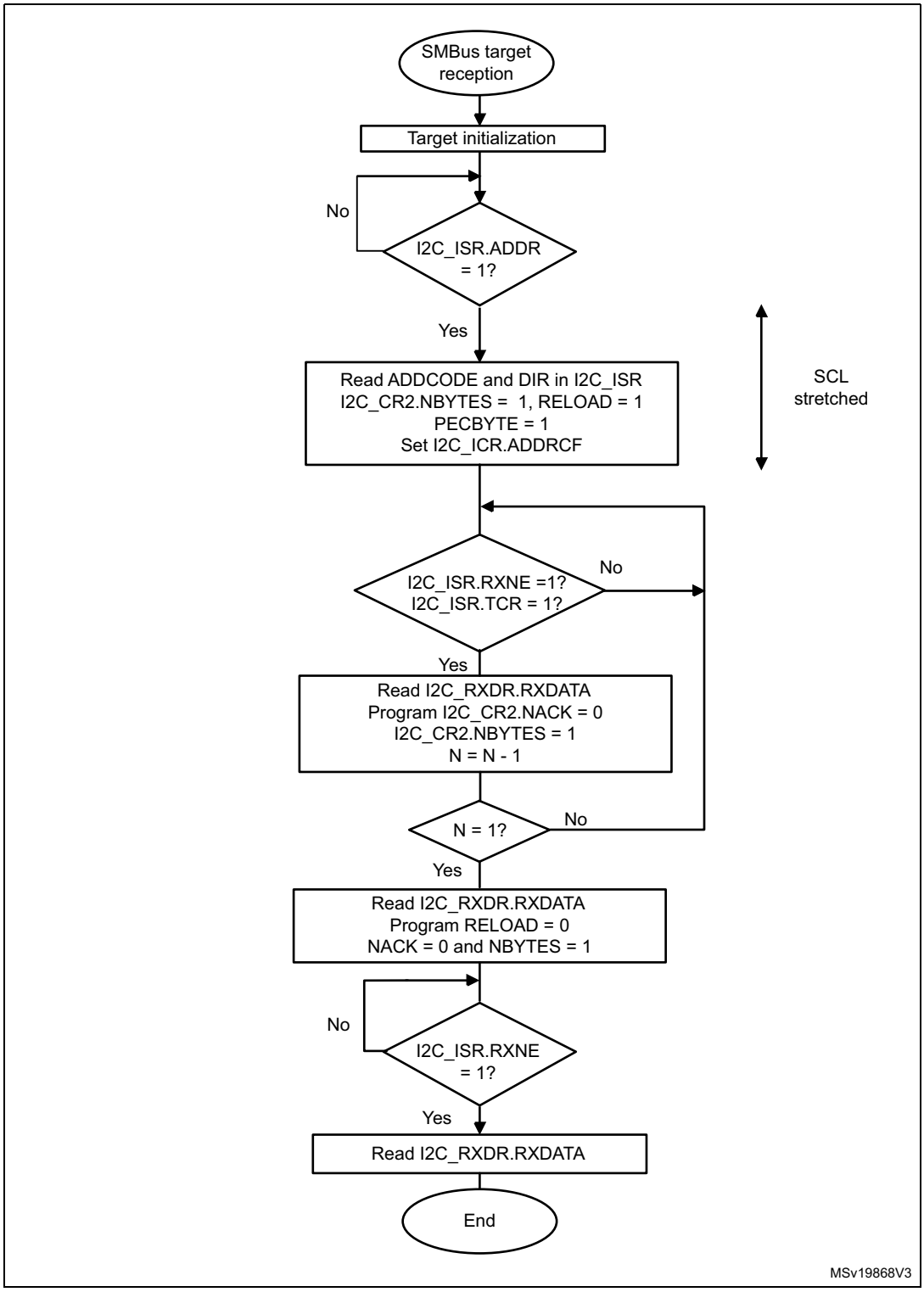
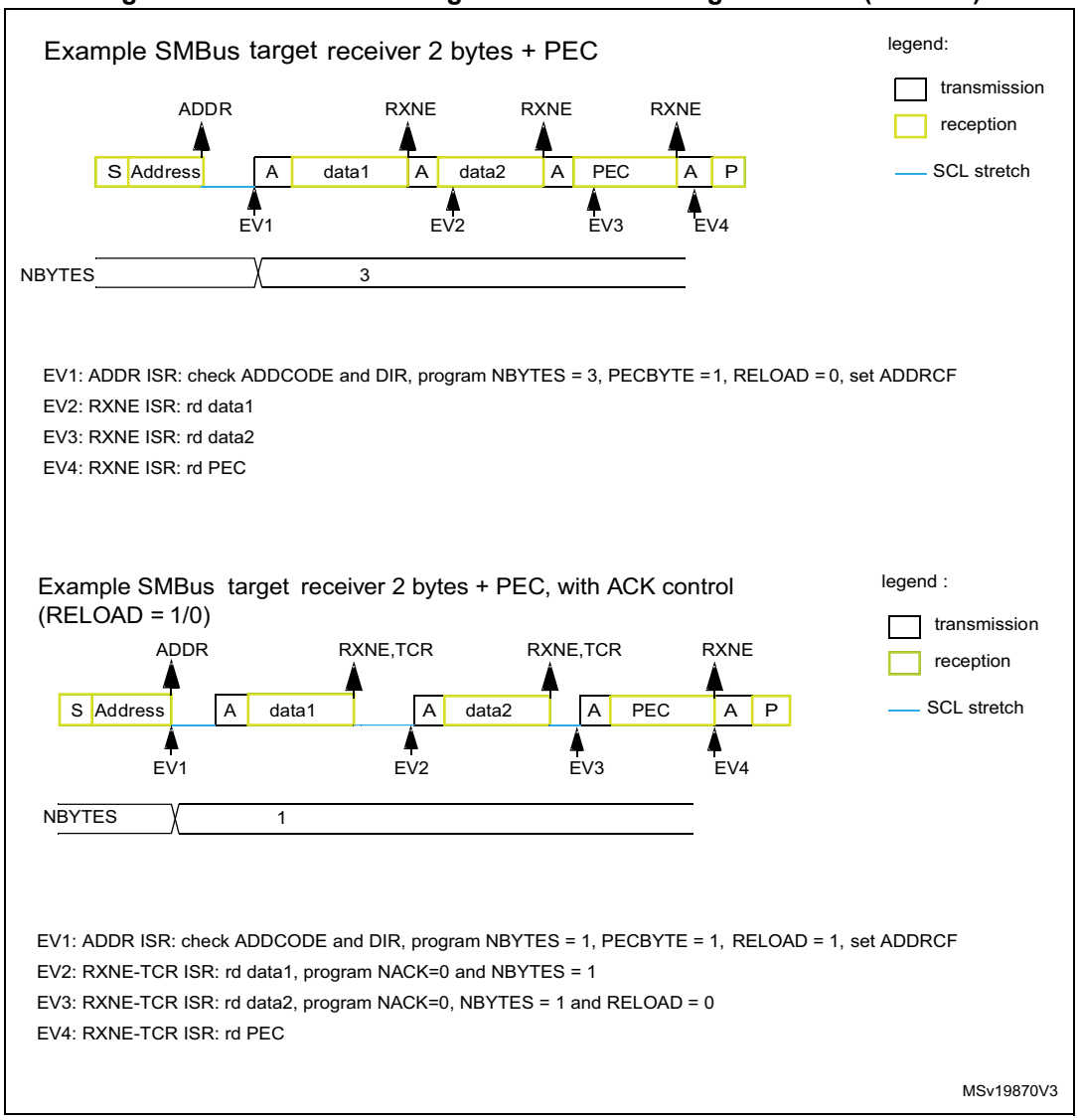


Figure 657. Bus transfer diagrams for SMBus target receiver (SBC = 1)



48.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 48.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

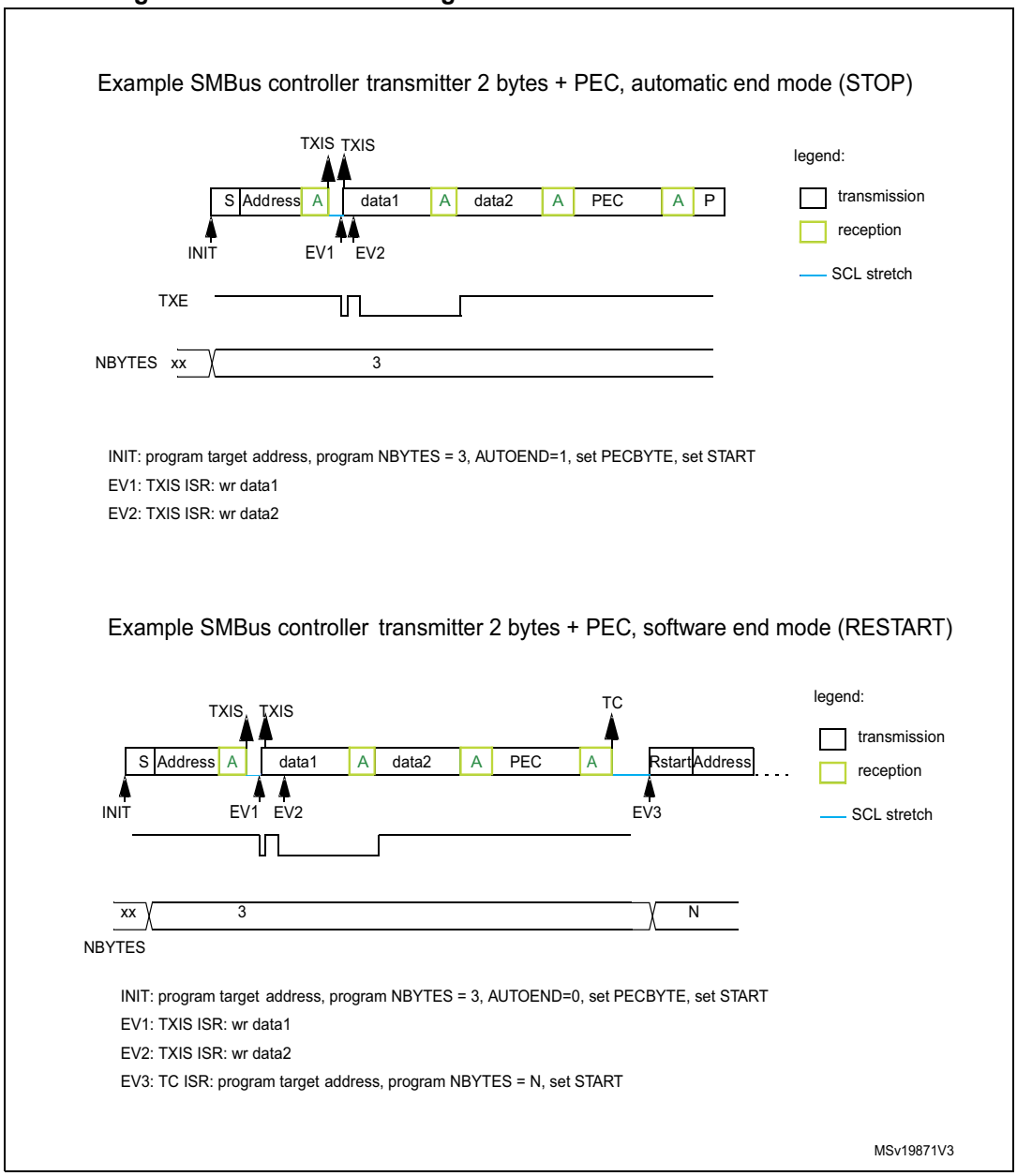
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 658. Bus transfer diagrams for SMBus controller transmitter



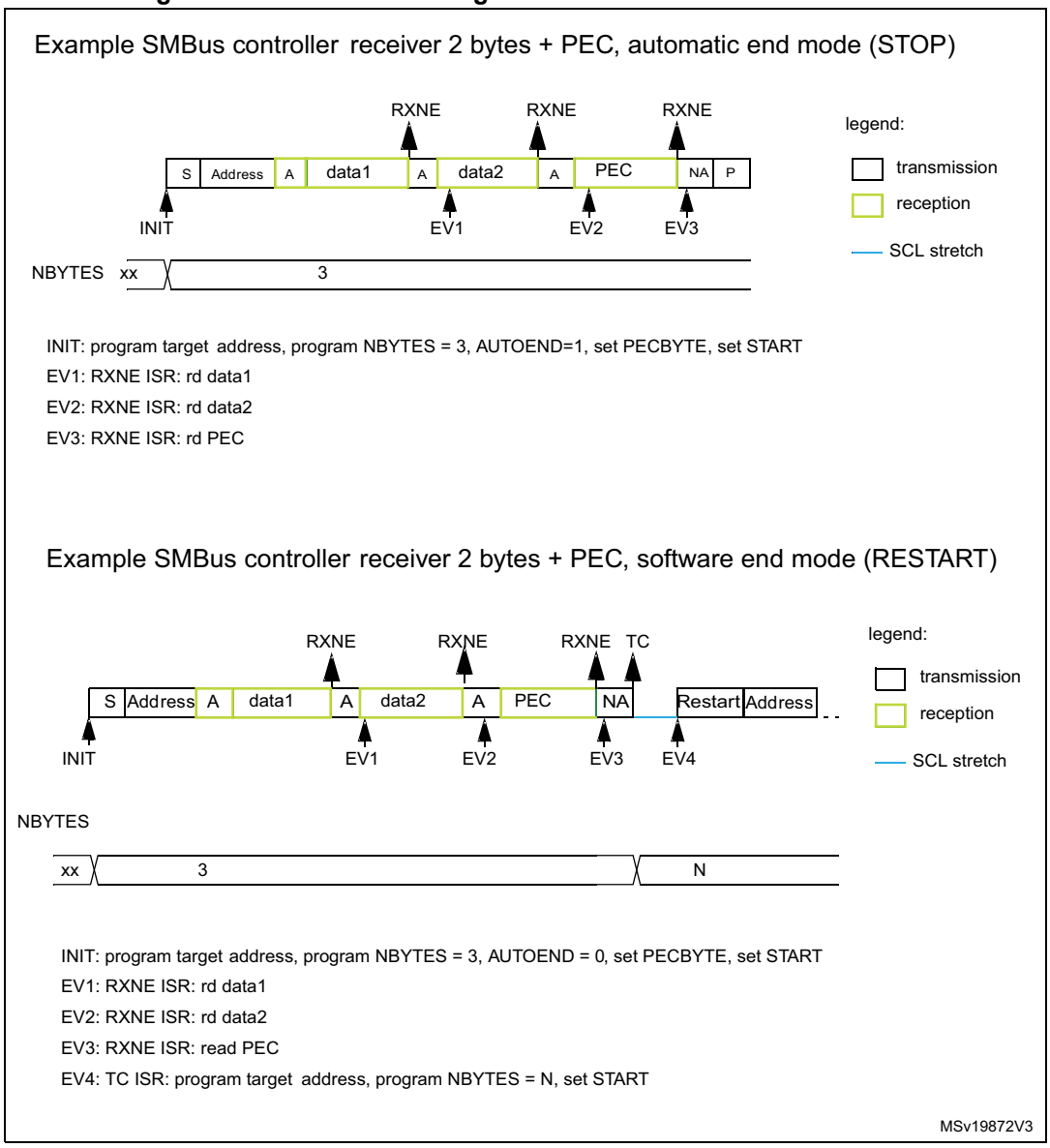
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 659. Bus transfer diagrams for SMBus controller receiver



48.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI and CSI only oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI and CSI only oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI and CSI only oscillator and stretches SCL low until the oscillator wakes up.

HSI and CSI only is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI and CSI only oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI and CSI only oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: The feature is only available when the HSI and CSI only oscillator is selected as the I2C clock.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

48.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

48.5 I2C in low-power modes

Table 530. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI and CSI only). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: the I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 48.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

48.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 531. I2C interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop modes	Exit Standby modes
I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
	Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
	STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
	Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
	Transfer complete	TC		Write START = 1 or STOP = 1			
	Address matched	ADDR	ADDRIE	Write ADDRCF = 1		Yes(1)	
	NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
	Arbitration loss	ARLO		Write ARLOCF = 1			
	Overrun/underrun	OVR		Write OVRCF = 1			
I2C_ERR	PEC error	PECERR	ERRIE	Write PECERRCF = 1	Yes	No	No
	Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF = 1			
	SMBus alert	ALERT		Write ALERTCF = 1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 48.3: I2C implementation*.

48.7 I2C DMA requests

48.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 16: General purpose direct memory access controller (GPDMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

48.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 16: General purpose direct memory access controller (GPDMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

48.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

48.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

48.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear

0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.

1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear

0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.

1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus 20 mA drive enable

0: 20 mA I/O drive disabled

1: 20 mA I/O drive enabled

Bit 23 **PECEN:** PEC enable

0: PEC calculation disabled

1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).

1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable
0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable
0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable
0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.
Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control
This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}
0000: Digital filter disabled
0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}
...
1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}
Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable
0: Error detection interrupts disabled
1: Error detection interrupts enabled
Note: Any of these errors generates an interrupt:
- arbitration loss (ARLO)
- bus error detection (BERR)
- overrun/underrun (OVR)
- timeout detection (TIMEOUT)
- PEC error detection (PECERR)
- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable
0: Transfer complete interrupt disabled
1: Transfer complete interrupt enabled
Note: Any of these events generates an interrupt:
Transfer complete (TC)
Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable
0: STOP detection (STOPF) interrupt disabled
1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable
0: Not acknowledge (NACKF) received interrupts disabled
1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)
0: Address match (ADDR) interrupts disabled
1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable
0: Receive (RXNE) interrupt disabled
1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable
0: Transmit (TXIS) interrupt disabled
1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable
0: Peripheral disabled
1: Peripheral enabled
Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

48.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer

1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.

This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

48.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.
10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

48.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is

completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable

0: Own address 2 disabled. The received target address OA2 is NACKed.

1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks

000: No mask

001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.

010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.

011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.

100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.

101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.

110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.

111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

Note: These bits can be written only when OA2EN = 0.

As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

48.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

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

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

48.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

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

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

– Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

48.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

48.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

48.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

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

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

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

48.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

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

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

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

48.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

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

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

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

48.9.12 I2C register map

The table below provides the I2C register map and the reset values.

Table 532. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

49 Improved inter-integrated circuit (I3C)

49.1 I3C introduction

The I3C interface handles communication with other devices such as sensor(s) and host processor.

An I3C bus is a two-wire, serial single-ended, multi-drop bus, intended to improve the legacy I^2C bus. The I3C interface improves the features of the I^2C interface (adding especially in-band interrupts, dynamic address assignment, energy-efficient data rates, power management) meanwhile preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C standard-mode (Sm), fast-mode (Fm), or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C SDR-only peripheral implements all the required features and many optional features of the MIPI® I3C specification v1.1 (see *Table 536: I3C peripheral controller/target features versus MIPI v1.1*). It can control all I3C bus-specific sequencing, protocol, arbitration, and timing.

After reset, the software can initialize the I3C peripheral as primary controller or as target:

- As I3C controller, the software can enable the I3C peripheral and initialize the I3C bus.
- As I3C target, the software can enable the I3C peripheral to operate on an I3C bus.

The I3C peripheral can be used with DMA, to off-load the CPU.

49.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in *Table 536*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages to legacy I^2C targets in Sm/Fm/Fm+
  - I3C SDR read/write private messages
  - I3C SDR broadcast CCC messages (see details in *Table 542*)
  - I3C SDR read/write direct CCC messages (see details in *Table 542*)

- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four I3C targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 49.3.2*):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 49.3.2*):
  - On a reset pattern
  - On a missed start
- Multi-clock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 49.6.2*)

49.3 I3C implementation

49.3.1 I3C instantiation

There is a single I3C instance in the device.

49.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 533*. For more details about the wake-up capabilities, refer to *Section 49.13*.

Table 533. I3C wake-up

Wake-up
From Stop mode with SVOS3

49.3.3 I3C FIFOs

The FIFOs are implemented as defined in *Table 534*.

Table 534. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

49.3.4 I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

49.3.5 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 535*.

Table 535. I3C interrupt(s)

Signal name	Signal type	Description
i3c_err_it	O	Error interrupt line
i3c_evt_it	O	Event interrupt line

49.3.6 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 536*.

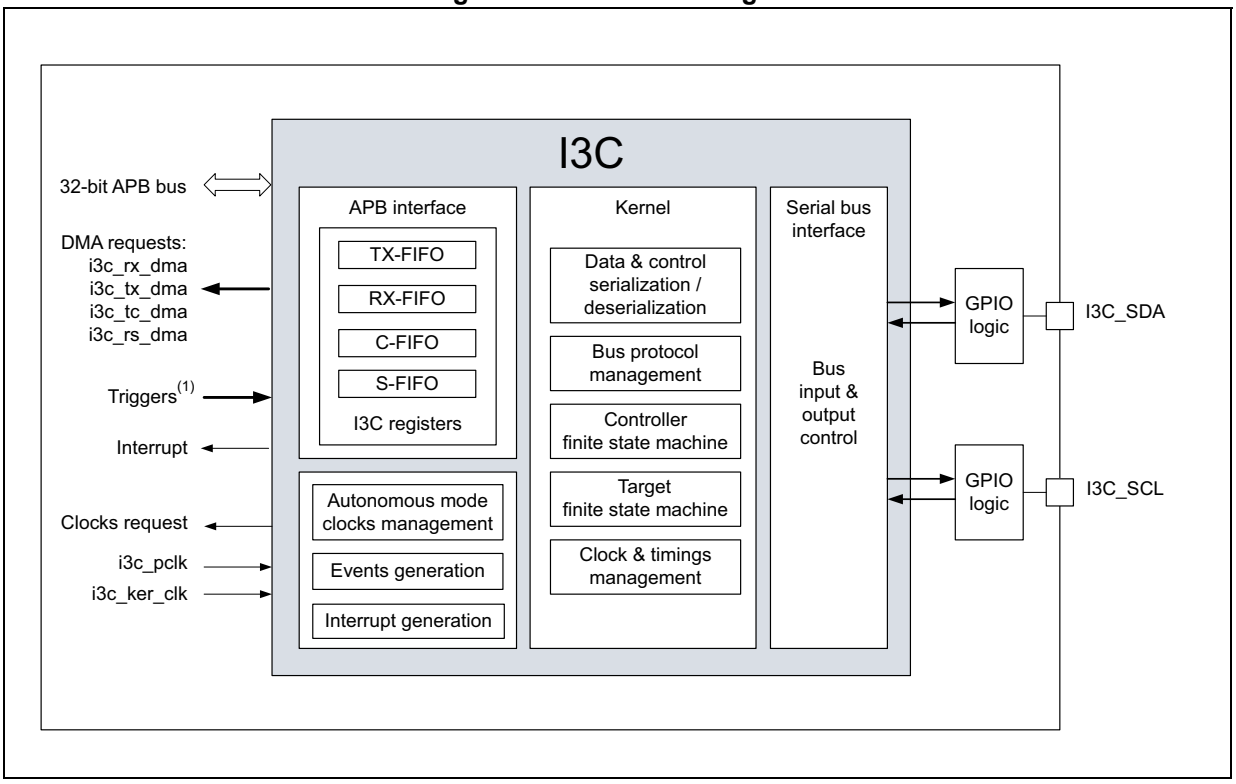
Table 536. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Sm/Fm/Fm+)	X	X	-	Mandatory when controller if the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	Supported broadcast ENTDAA when controller and target. Supported broadcast SETAASA and direct SETDASA when controller.
7-bit static I^2C address	X	X	-	No support of the I3C peripheral as target on an I^2C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 542* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller. Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

49.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 660*.

Figure 660. I3C block diagram



1. Refer to *Section 49.3.4: I3C triggers*.

49.5 I3C pins and internal signals

Table 537. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	Serial bus data line
I3C_SCL	Input/output	Serial bus clock line

Table 538. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line

Table 538. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller

1. Refer to *Section 49.3.5: I3C interrupt(s)*.

49.6 I3C reset and clocks

49.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 542* and *Section 49.16.16* for more details.

This reset interrupt notification can be used to wake up from a low power mode. For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

49.6.2 I3C clocks and requirements

As indicated in *Figure 660*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE} for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to section Reset and clock control (RCC) for more details).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

Note:

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

49.7 I3C peripheral state and programming

49.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 661* and *Figure 662*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different initialized configuration registers, and switches to the (enabled and) idle state. The peripheral is initialized as I3C target or I3C controller.
  - As I3C controller, then software can enable the I3C peripheral and initialize the I3C bus.
  - As I3C target, then the software can enable the I3C peripheral to operate on an I3C bus.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

    As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

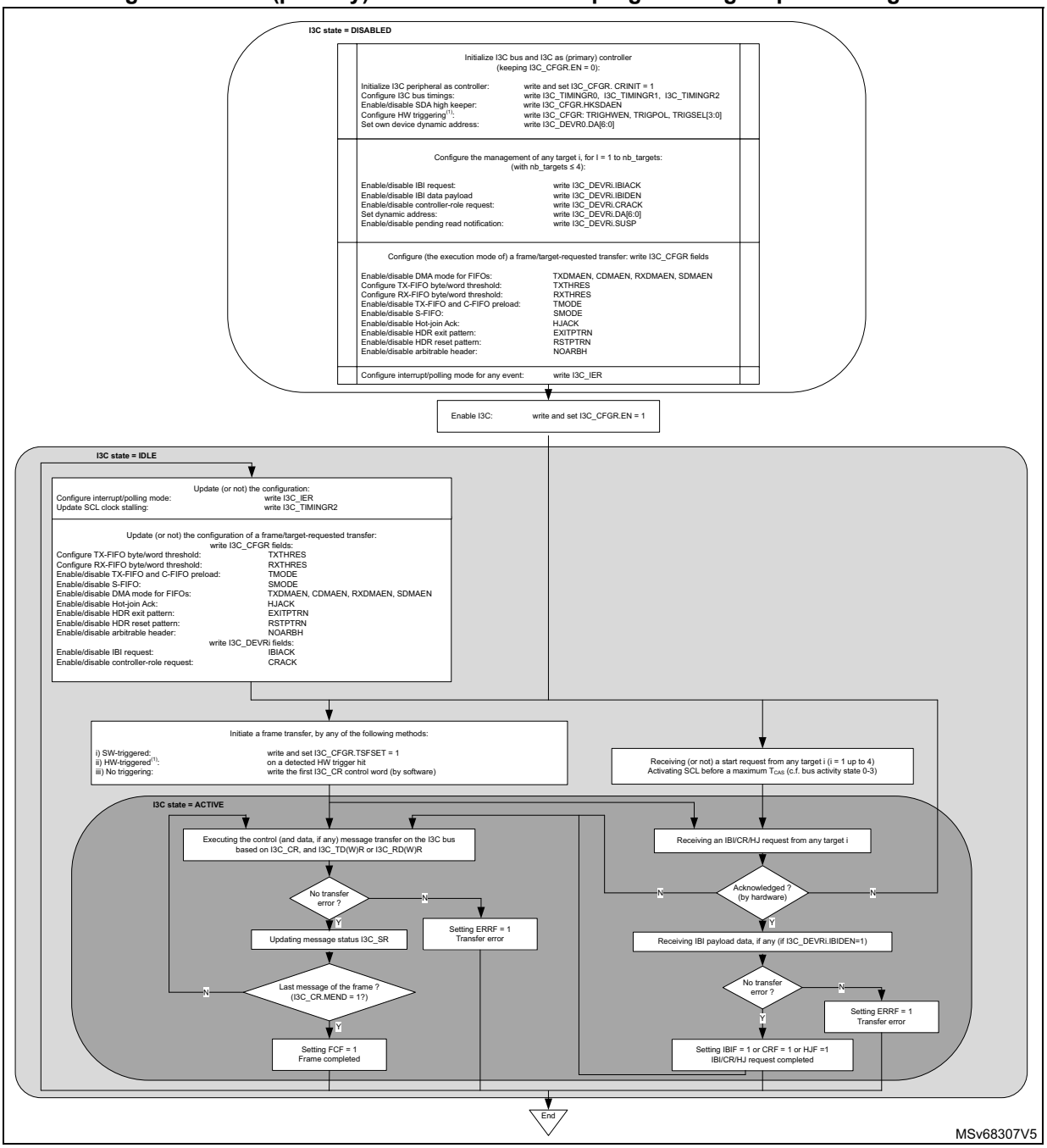
    As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF).

Note: The software can disable the I3C peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

49.7.2 I3C controller state and programming sequence

Figure 661 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 661. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 49.3.4: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:
  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG_H_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG_OD_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 49.9*:

- *Figure 673: IBI transfer, as controller/target*
- *Figure 674: Hot-join request transfer, as controller/target*
- *Figure 675: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 49.9*:

- *Figure 663: I3C CCC messages, as controller*
- *Figure 664: I3C broadcast ENTDAA CCC, as controller*
- *Figure 665: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 670: I3C private read/write messages, as controller*
- *Figure 672: Legacy I2C read/write messages, as controller*

Figure 661 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 49.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 49.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

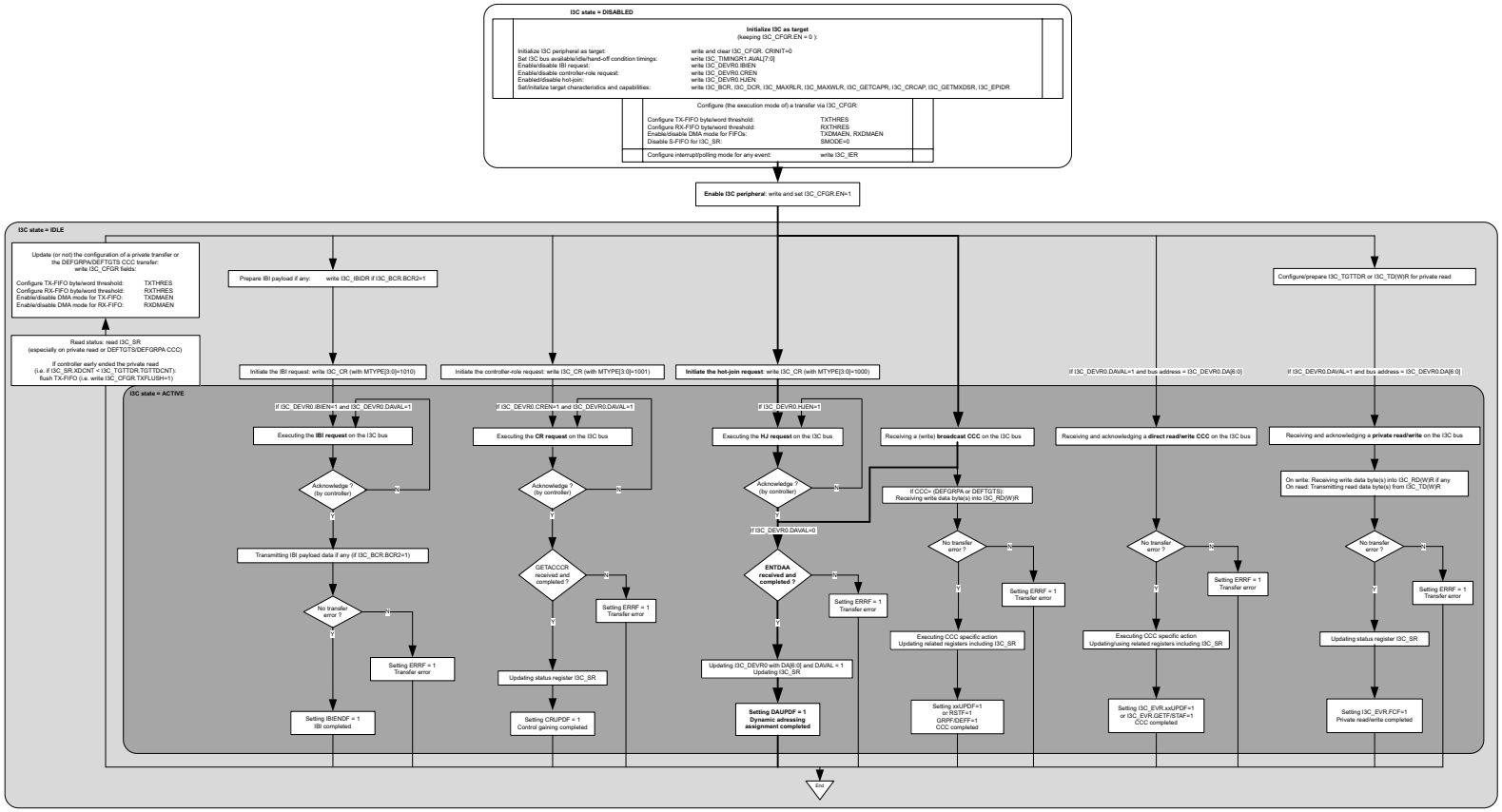
The registers usage versus the I3C peripheral role as controller is summarized in *Section 49.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 540*.

49.7.3 I3C target state and programming sequence

Figure 662 illustrates the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 662. I3C target state and programing sequence diagram



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  - Bus available condition time (t_{AVAL}) for IBI or controller-role request
  - Bus idle condition time (t_{IDLE}) for hot-join request
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - *I3C bus characteristics register (I3C_BCR)*
  - *I3C device characteristics register (I3C_DCR)*
  - *I3C maximum read length register (I3C_MAXRLR)*
  - *I3C maximum write length register (I3C_MAXWLR)*
  - *I3C get capability register (I3C_GETCAPR)*
  - *I3C controller-role capability register (I3C_CRCAPR)*
  - *I3C get max data speed register (I3C_GETMXDSR)*
  - *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message on the I3C bus

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 49.9*:

- *Figure 666: I3C CCC messages, as target*
- *Figure 667: I3C broadcast ENTDAA CCC, as target*
- *Figure 668: I3C broadcast DEFTGTS CCC, as target*
- *Figure 669: I3C broadcast DEFGRPA CCC, as target*
- *Figure 671: I3C private read/write messages, as target*

Figure 662 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 49.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer on the I3C bus

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 49.9*:

- *Figure 673: IBI transfer, as controller/target*
- *Figure 674: Hot-join request transfer, as controller/target*
- *Figure 675: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC)

The registers usage vs. the I3C peripheral role as target is summarized in *Section 49.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 541*.

49.8 I3C registers and programming

49.8.1 I3C register set, as controller/target

Table 539 lists the registers and their usage versus the I3C peripheral role.

Table 539. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_DEVR0	X	X

Table 539. I3C register usage (continued)

Register	Used as controller	Used as target
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

49.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 540* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 540. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X

Table 540. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CFGR	X	CRINIT HKSDAEN EN(1)	Write: any used field except {CRINIT, HKSDAEN}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 540. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

49.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 541* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 541. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 541. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-

Table 541. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

49.9 I3C bus transfers and programming

49.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 542*.

Table 542. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF if DAVAL = 0	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 668*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	-
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X	-	-
SETAASA	0x29		No defining/sub-command byte	No data byte	X	-	-
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 669*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 542. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	-	-
SETNEWDA	0x88			With one data byte	X	X, DAUPDF if DAVAL = 1	Update I3C_DEVR0: DA[6:0] (and keep DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 49.16.19*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 49.16.18*.

Table 542. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 49.16.28*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 49.16.23*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 49.16.24*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 49.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 49.16.27*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 49.16.25*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 49.16.26*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

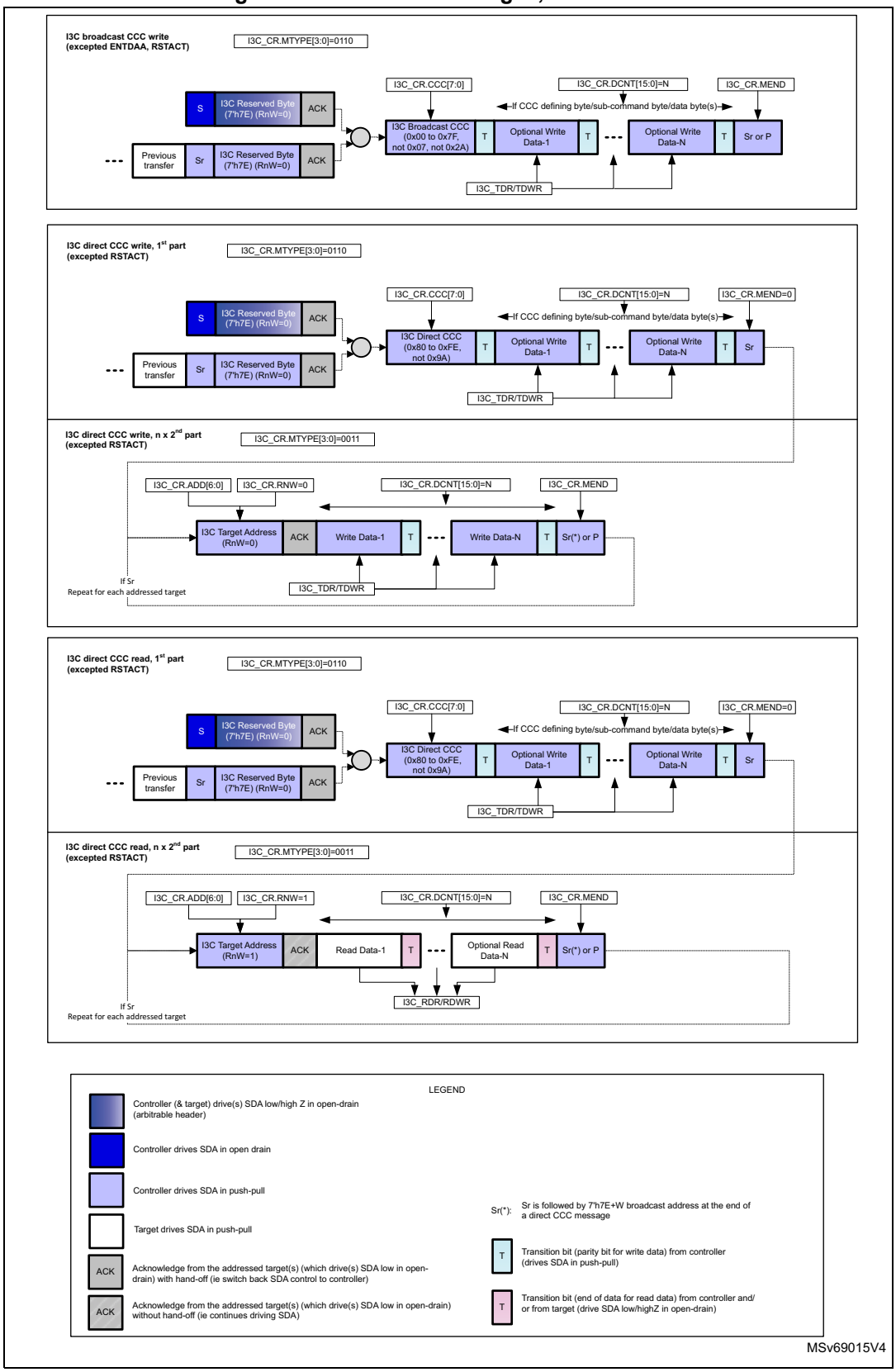
Table 542. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

49.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 663 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

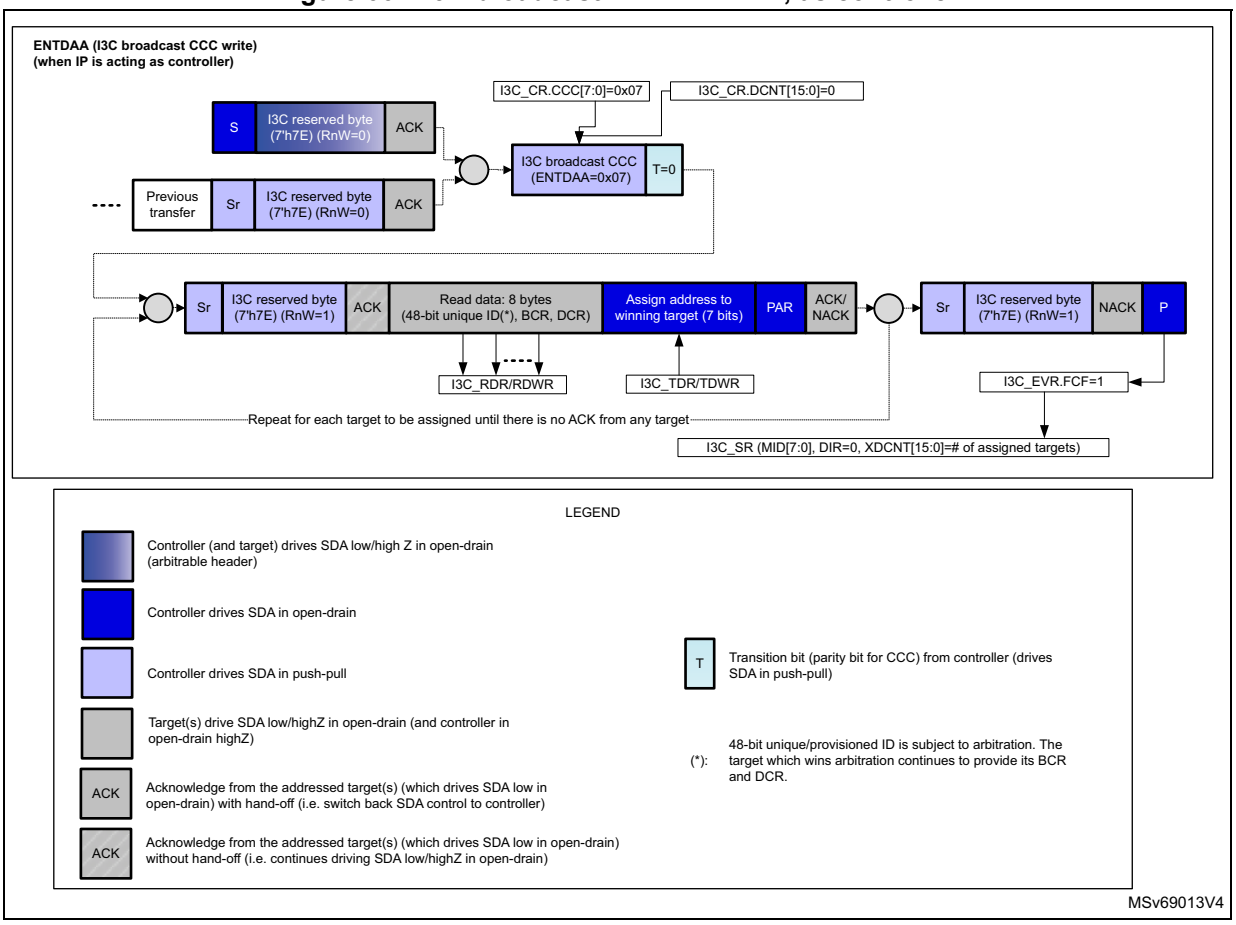
Figure 663. I3C CCC messages, as controller



49.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 664 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

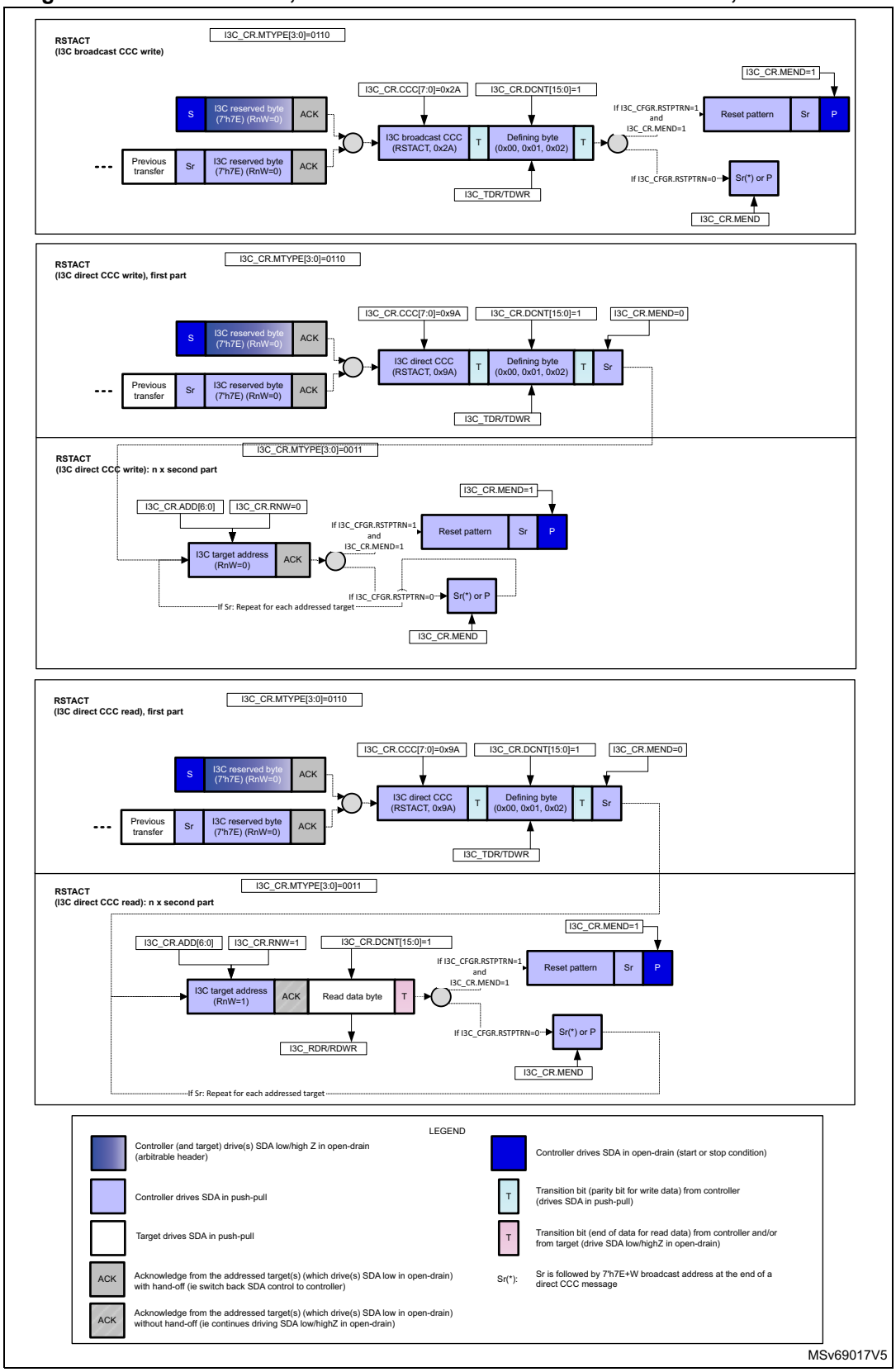
Figure 664. I3C broadcast ENTDAA CCC, as controller



49.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 665 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

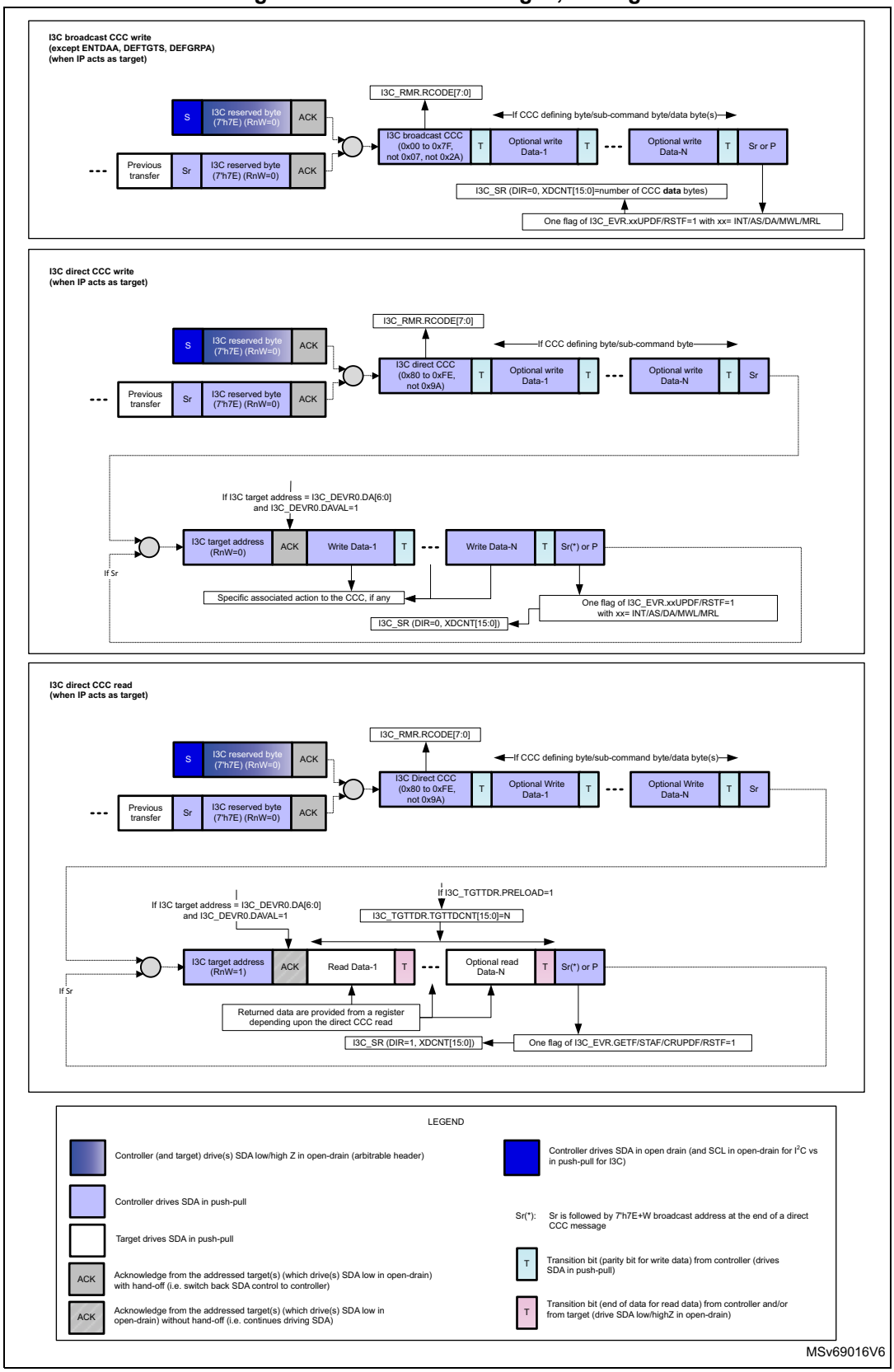
Figure 665. I3C broadcast, direct read and direct write RSTACT CCC, as controller



49.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 666 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

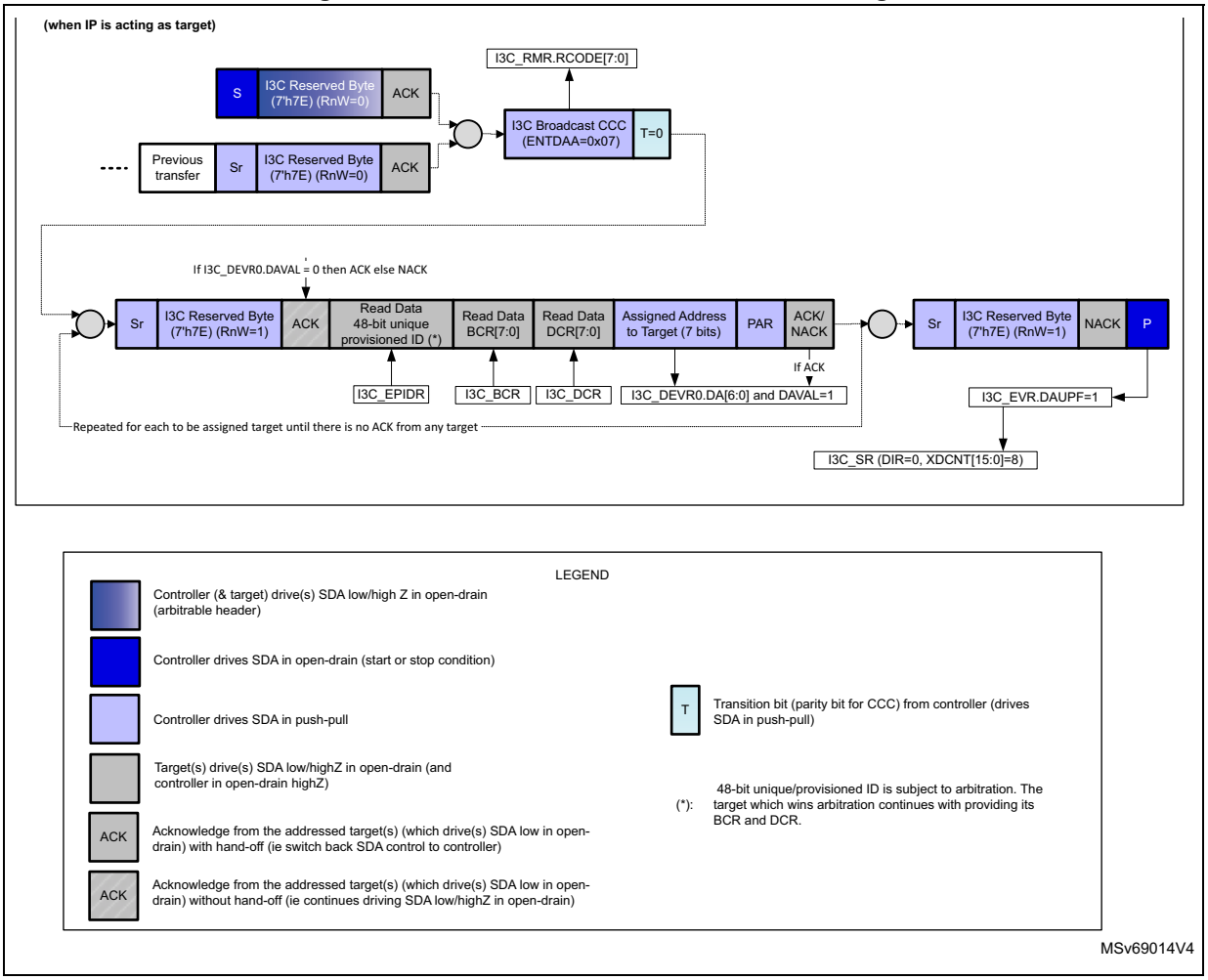
Figure 666. I3C CCC messages, as target



49.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 667 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

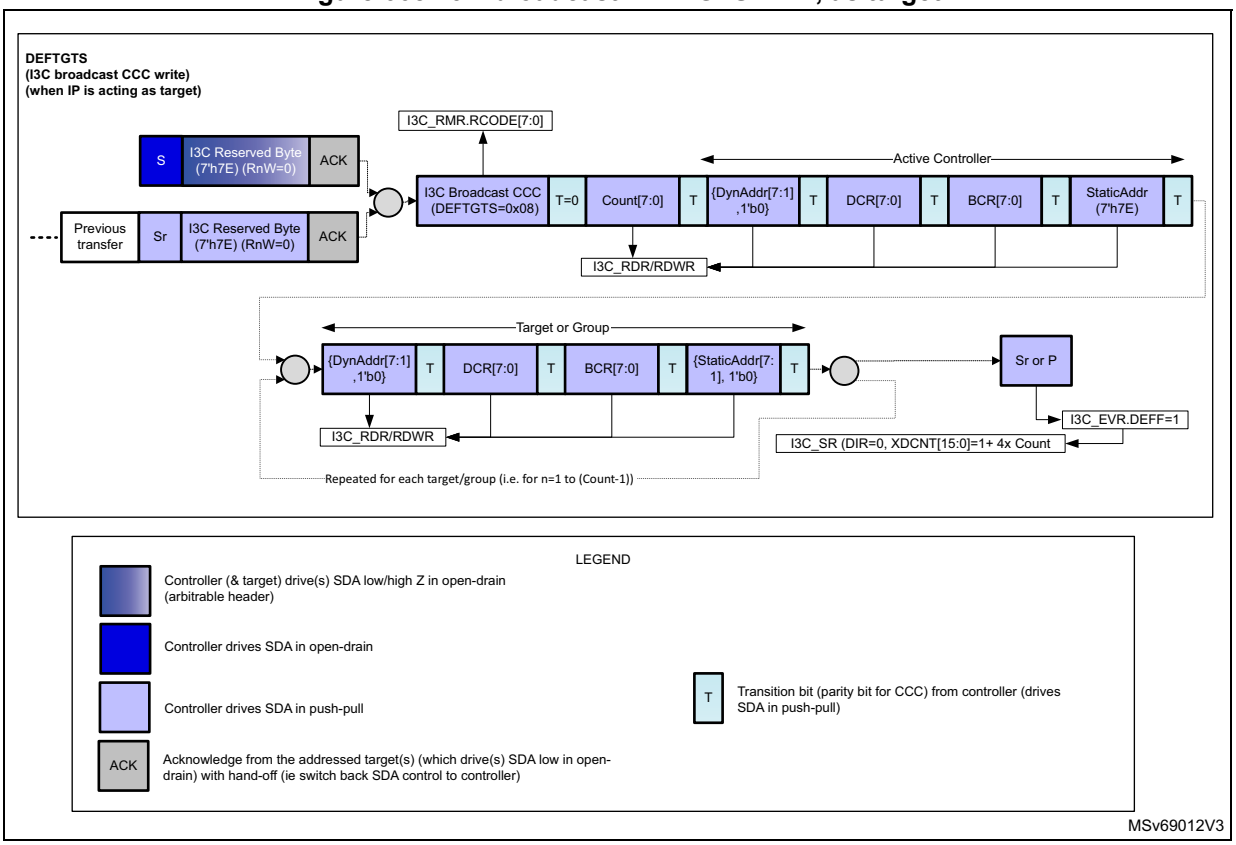
Figure 667. I3C broadcast ENTDAA CCC, as target



49.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 668 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

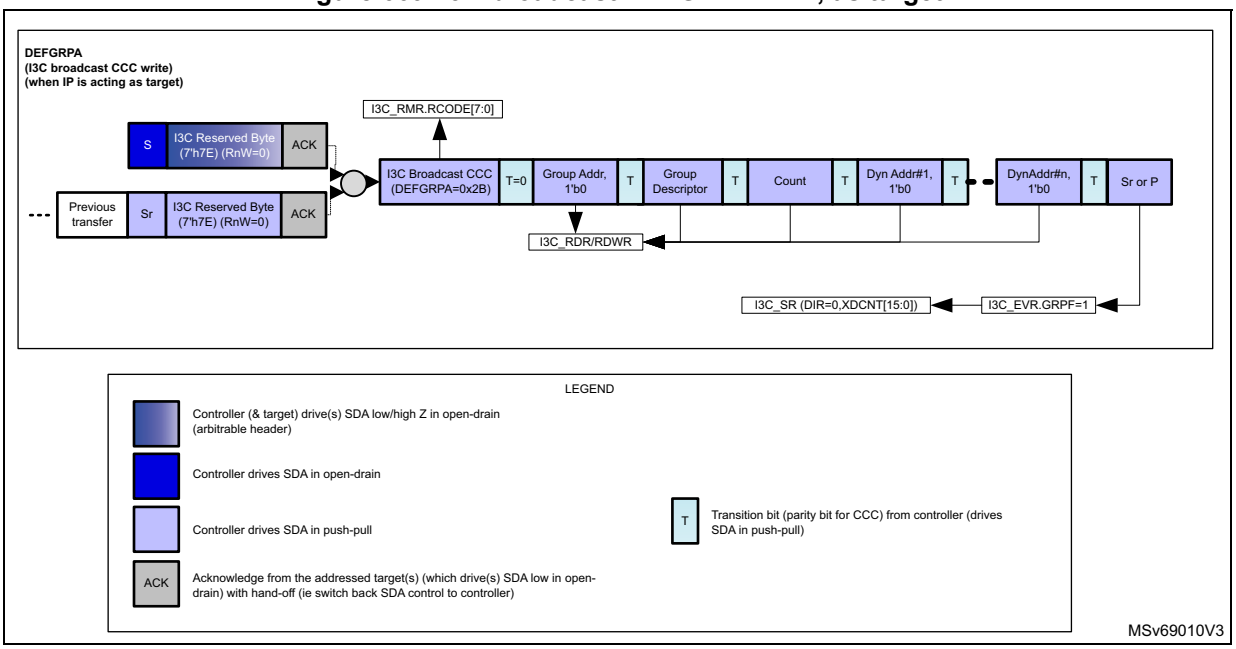
Figure 668. I3C broadcast DEFTGTS CCC, as target



49.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 669 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 669. I3C broadcast DEFGRPA CCC, as target



49.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

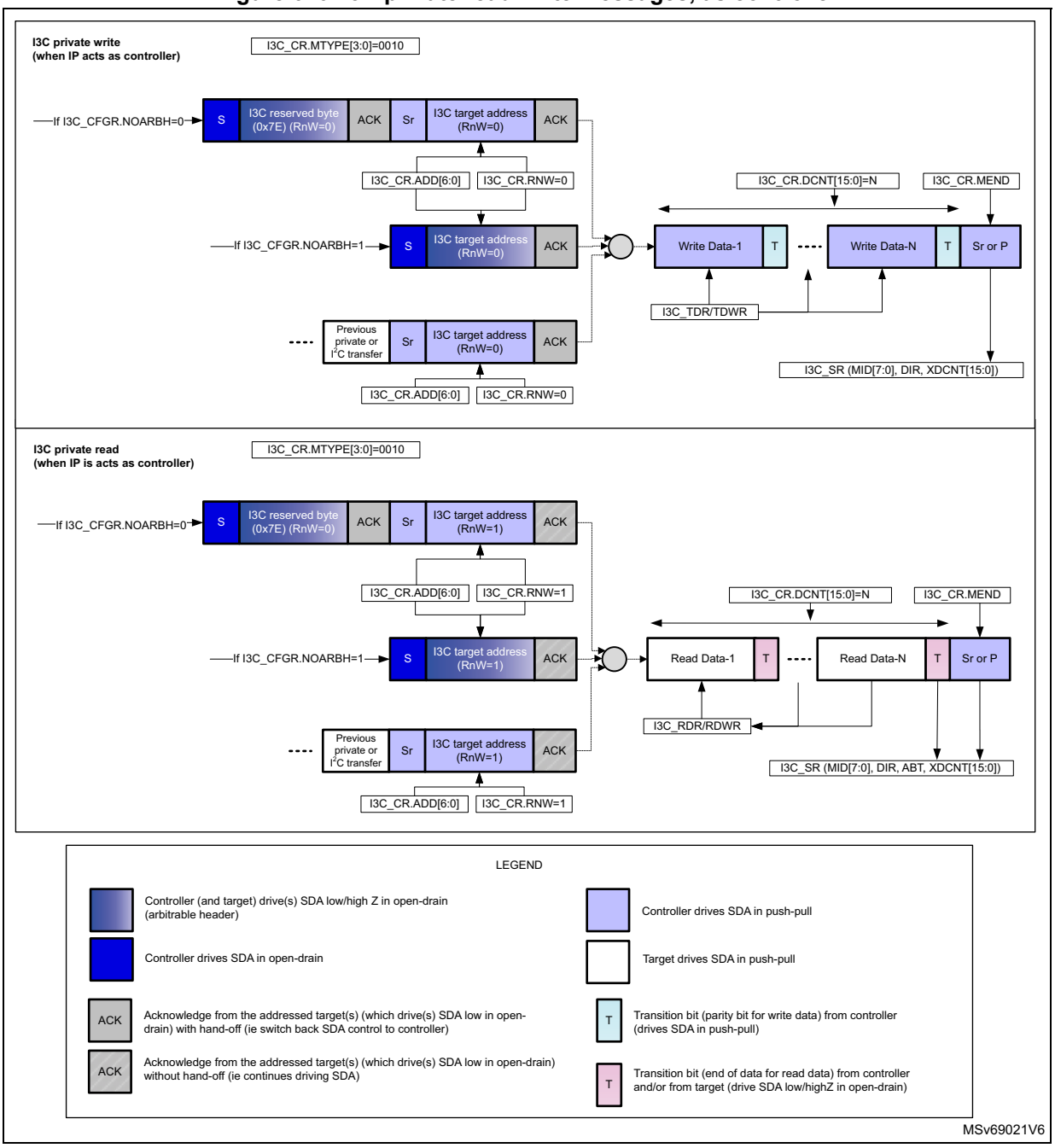
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

49.9.10 I3C private read/write transfer, as controller

Figure 670 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

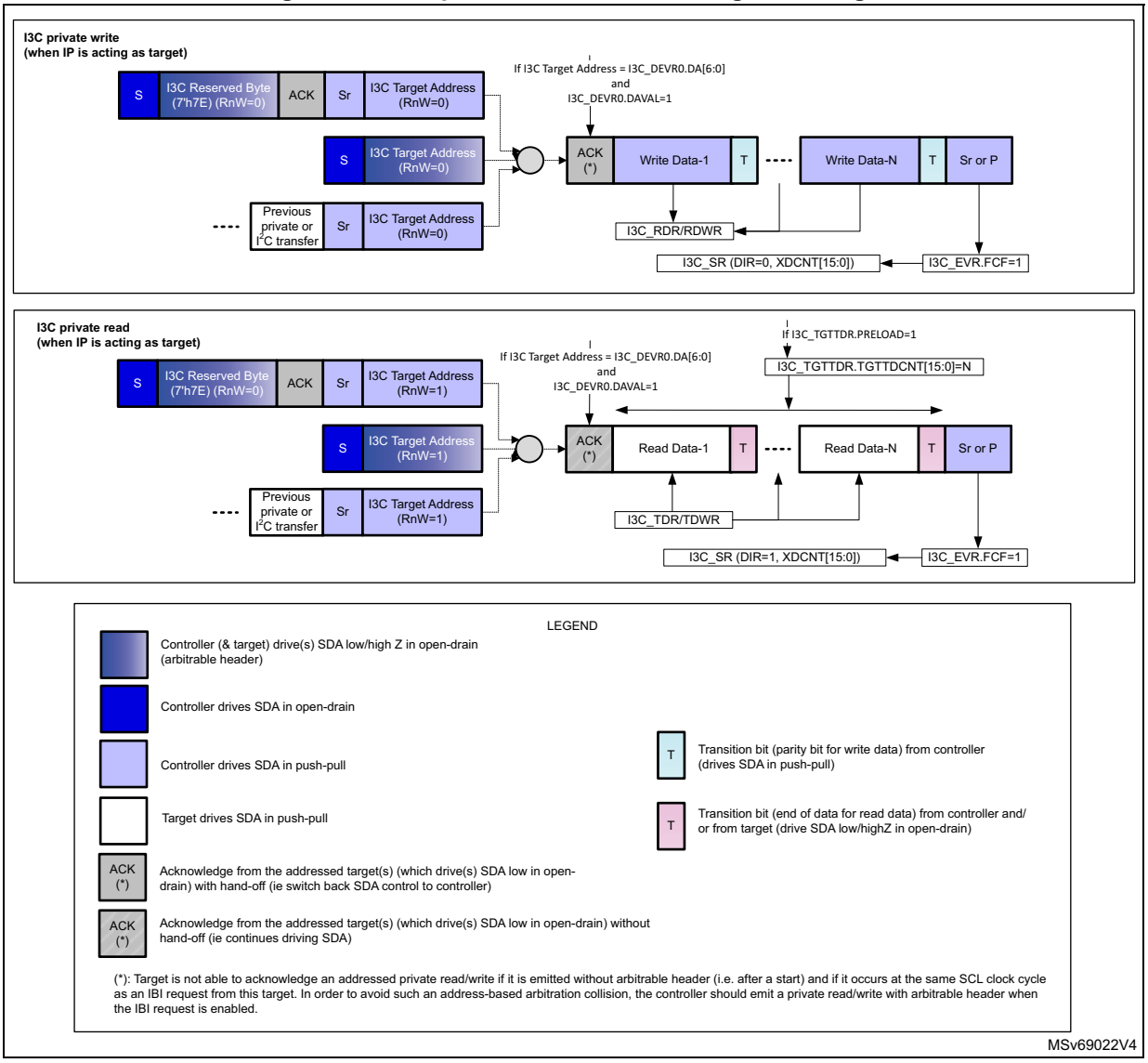
Figure 670. I3C private read/write messages, as controller



49.9.11 I3C private read/write transfer, as target

Figure 671 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

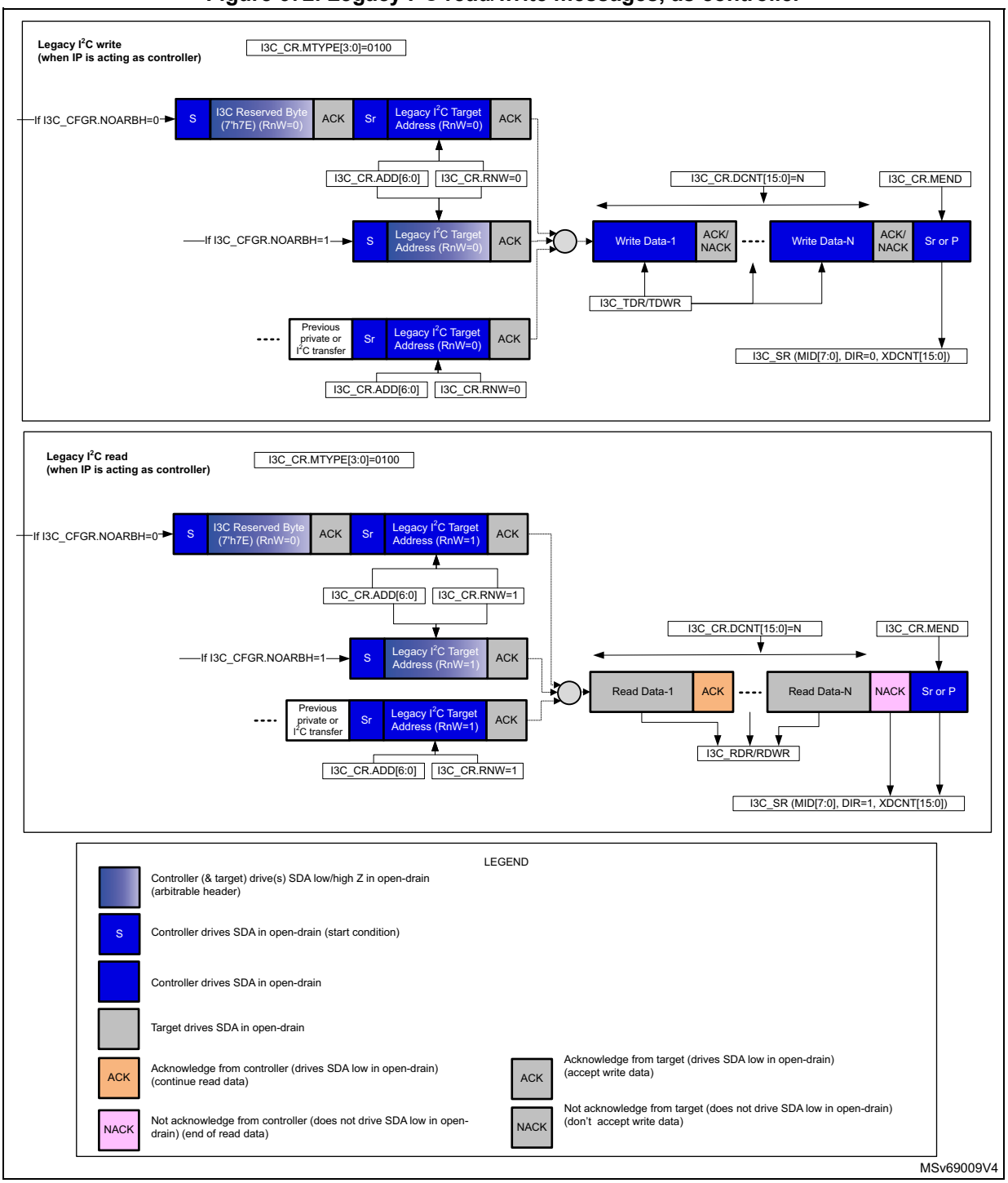
Figure 671. I3C private read/write messages, as target



49.9.12 Legacy I²C read/write transfer, as controller

Figure 672 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

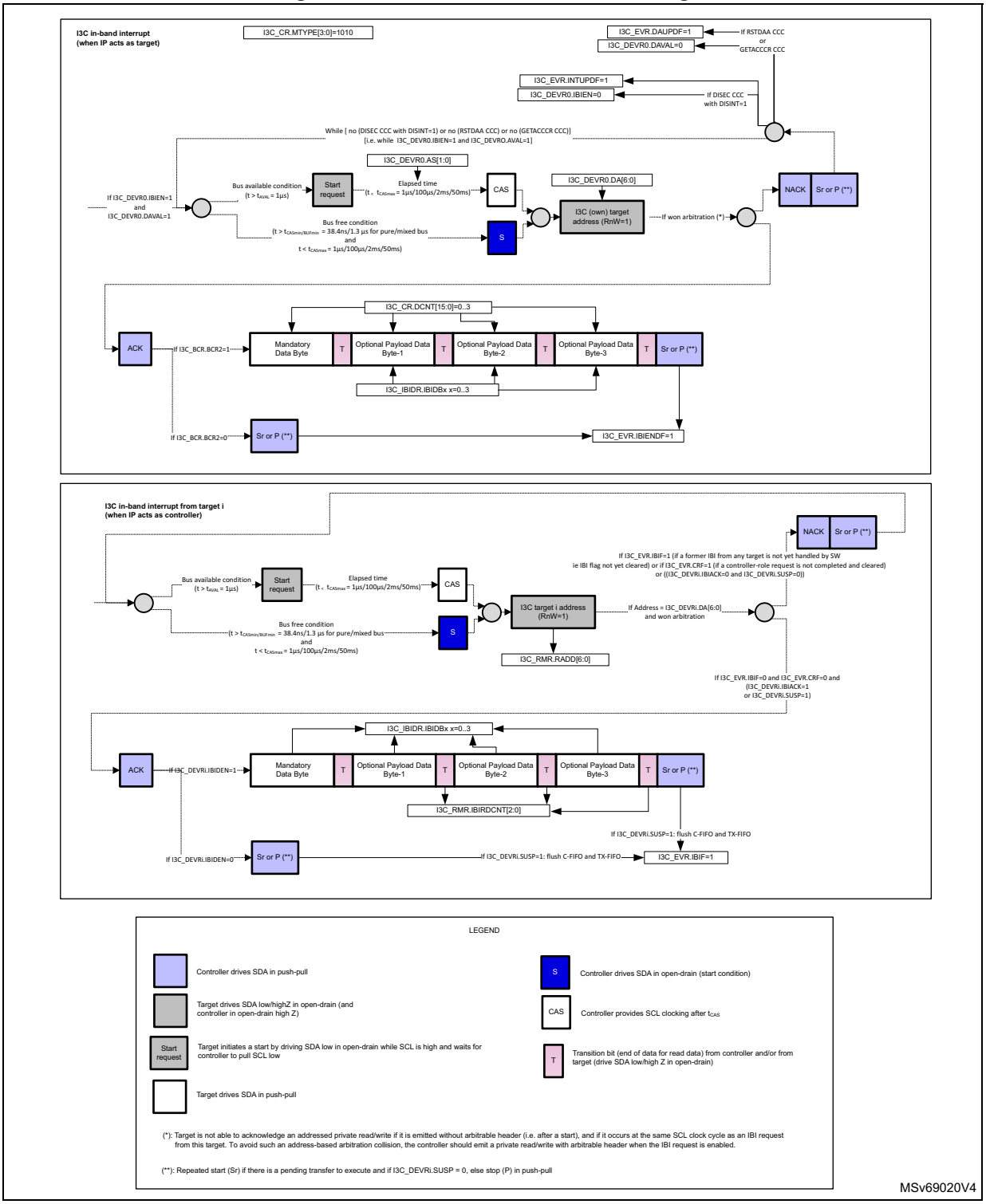
Figure 672. Legacy I²C read/write messages, as controller



49.9.13 I3C IBI transfer, as controller/target

Figure 673 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 673. IBI transfer, as controller/target

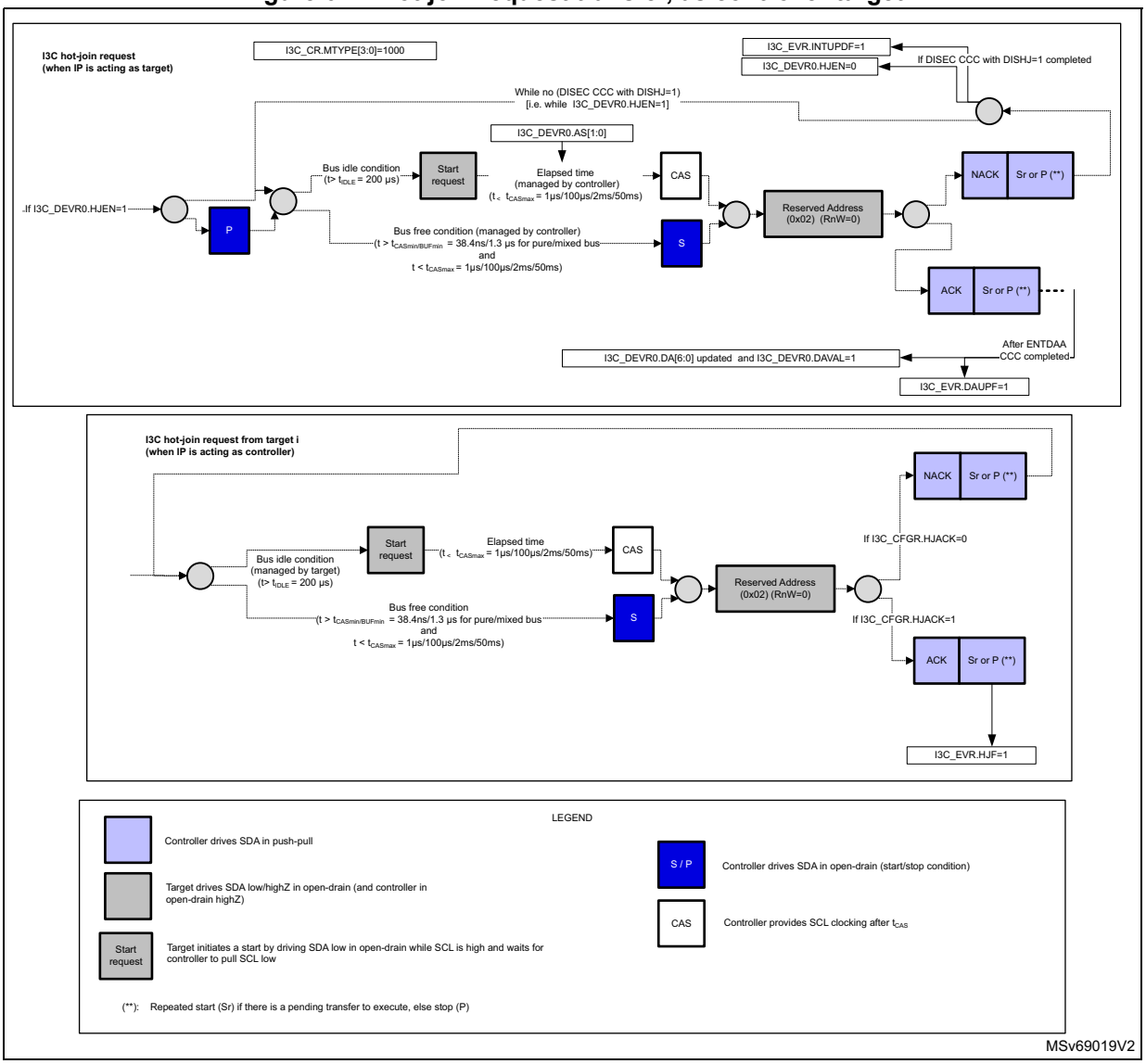


When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

49.9.14 I3C hot-join request transfer, as controller/target

Figure 674 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

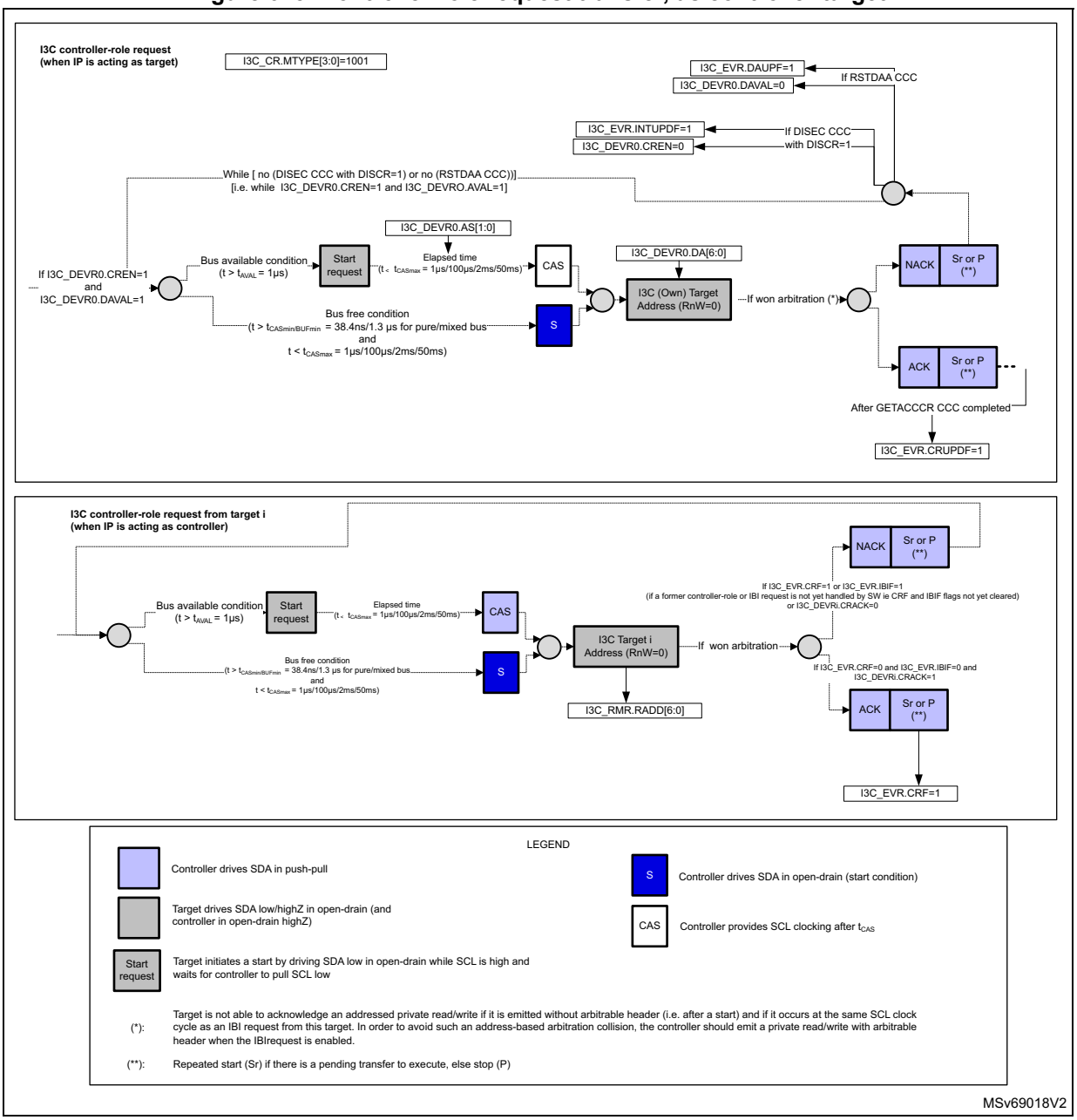
Figure 674. Hot-join request transfer, as controller/target



49.9.15 I3C controller-role request transfer, as controller/target

Figure 675 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 675. Controller-role request transfer, as controller/target



49.10 I3C FIFOs management, as controller

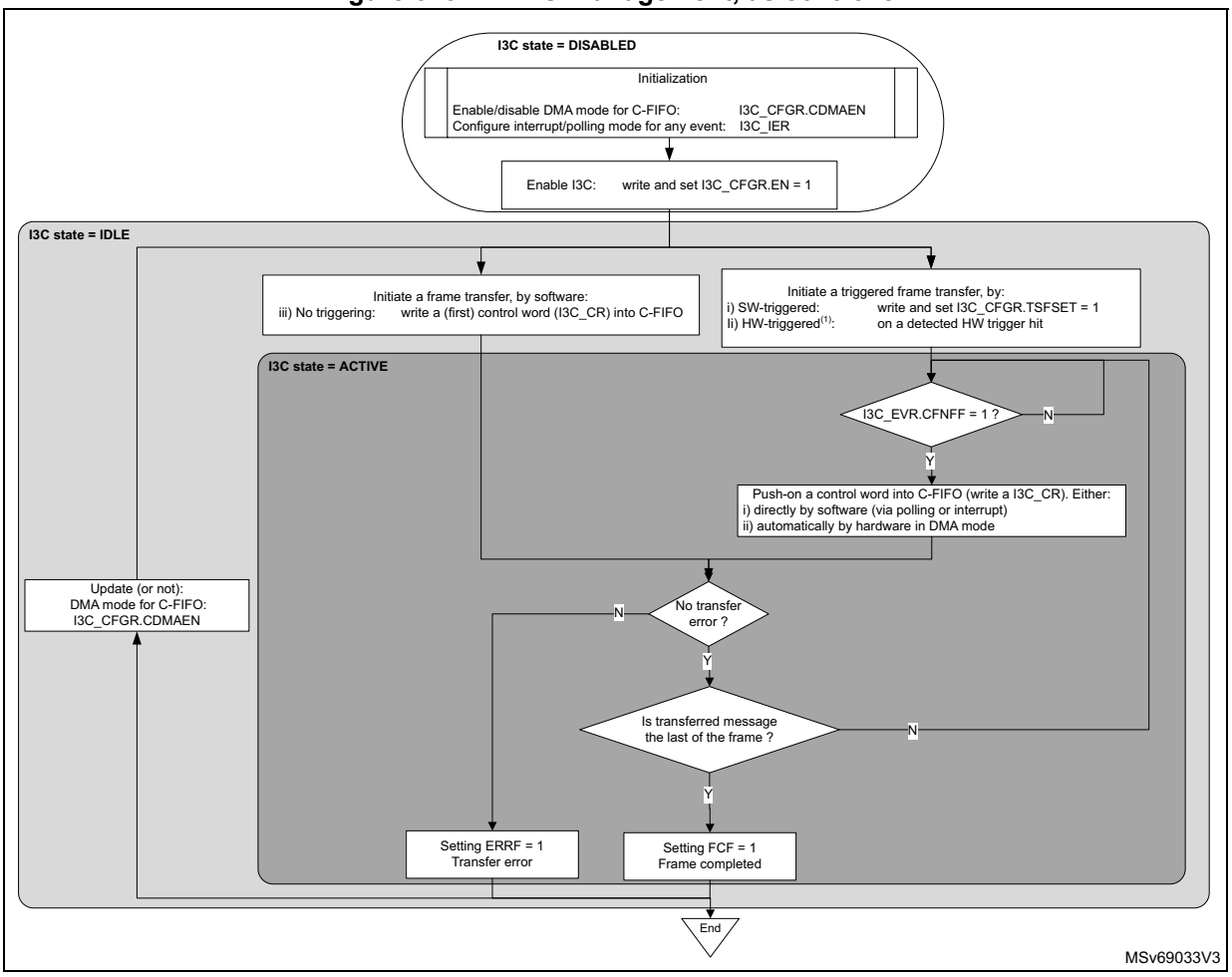
49.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 49.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 663*, *Figure 664*, *Figure 665*)
- direct read/write CCC (*Figure 665*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 670*)
- legacy I2C read/write (*Figure 672*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 676 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 676. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 49.3.4*.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

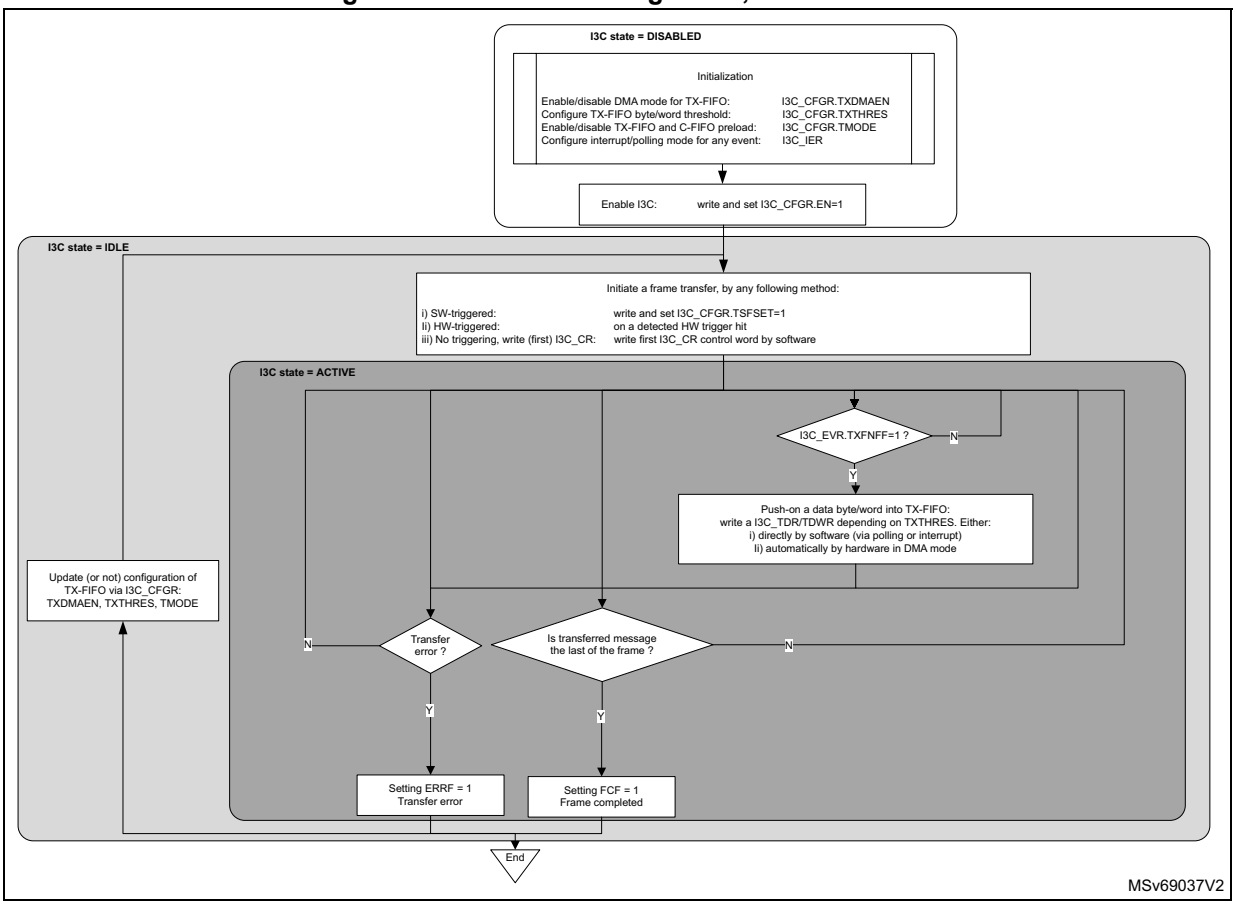
49.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 49.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 663*, *Figure 664*, *Figure 665*)
- private write (*Figure 670*)
- legacy I2C write (*Figure 672*)

Figure 677 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 677. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 534*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

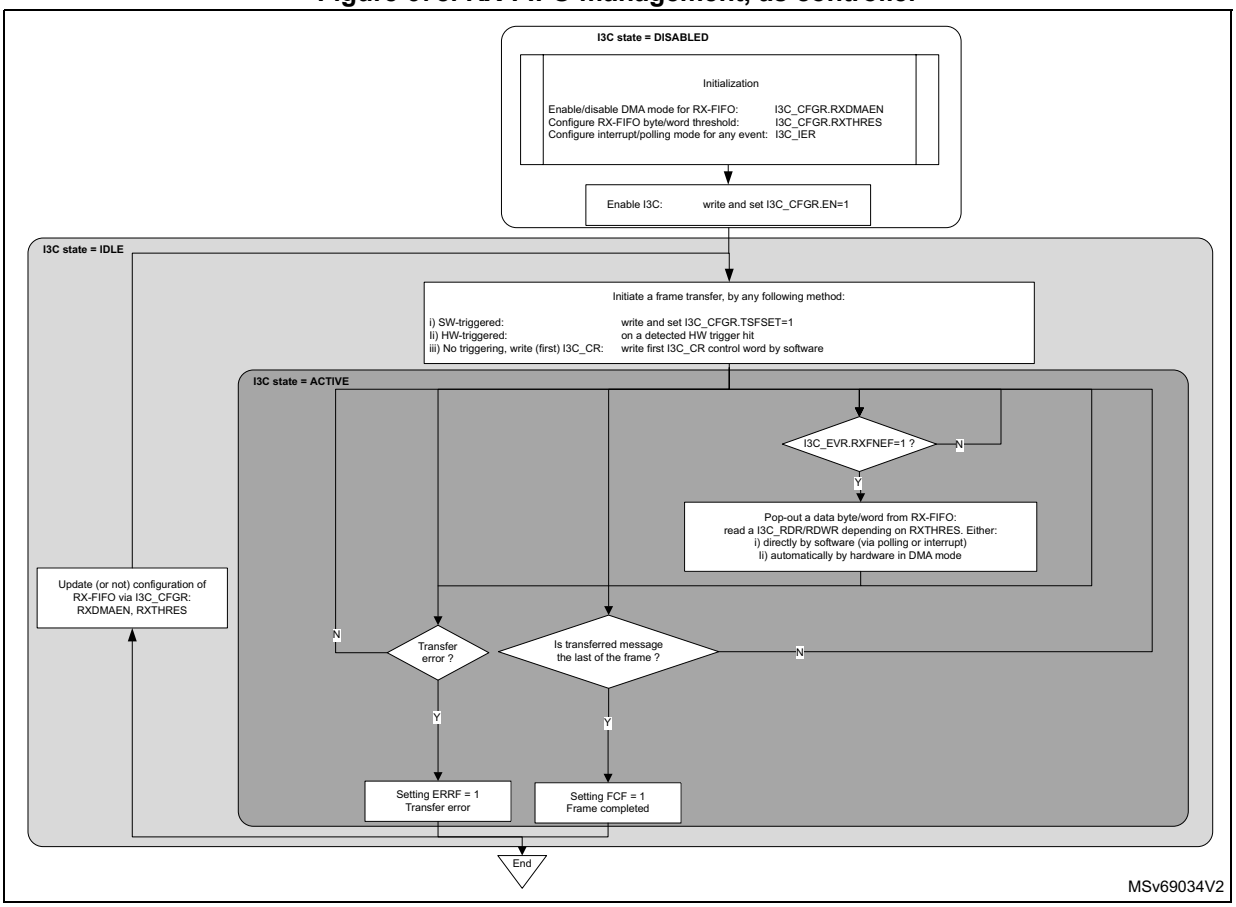
49.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 49.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 664*)
- direct CCC read (*Figure 663*), including direct RSTACT CCC read (*Figure 665*)
- private read (*Figure 670*)
- legacy I2C read (*Figure 672*)

Figure 678 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 678. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 49.10.4*.

49.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

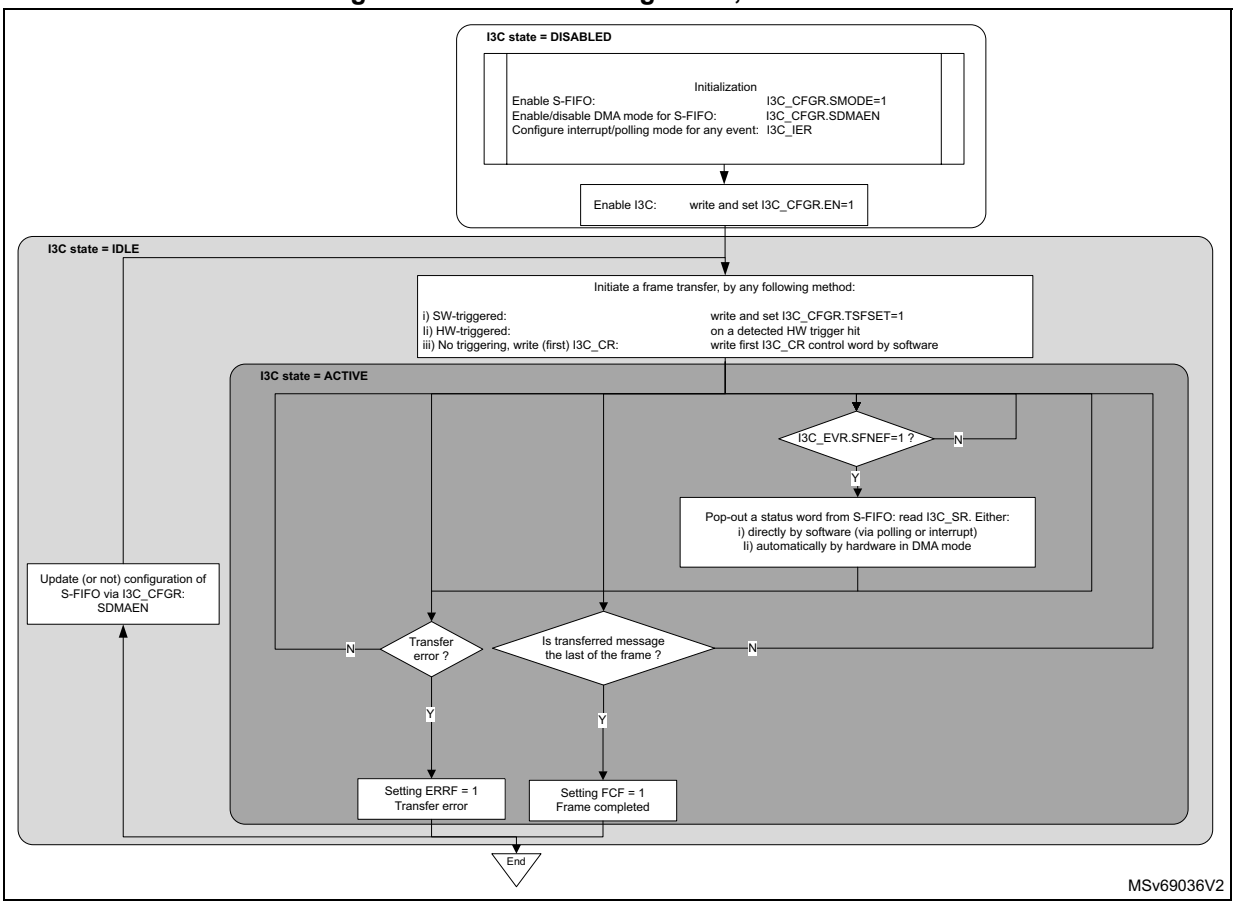
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 49.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 679* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 679. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

49.11 I3C FIFOs management, as target

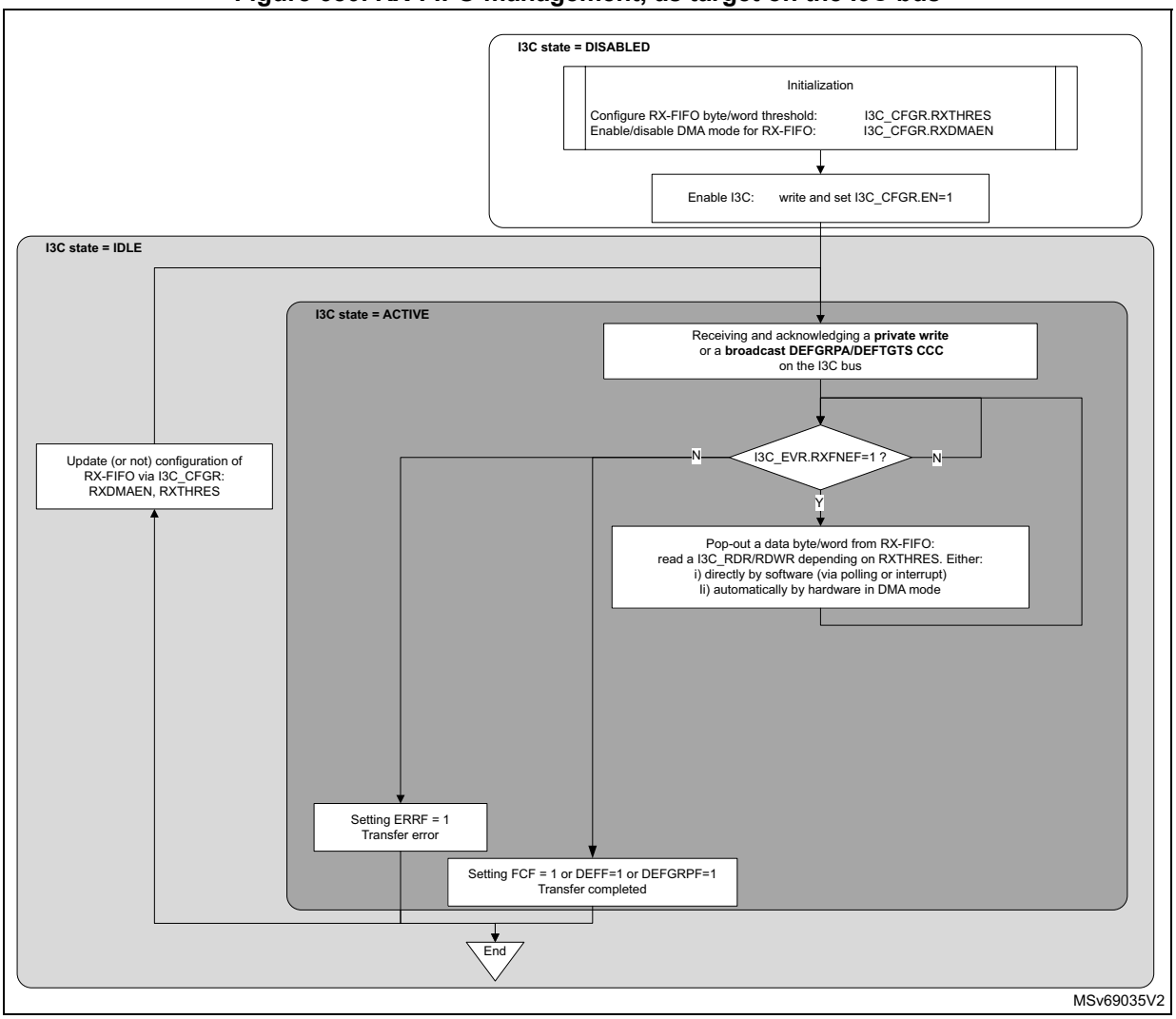
49.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 49.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 668*)
- broadcast DEFGRPA CCC (*Figure 669*)
- private write (*Figure 671*)

Figure 680 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received on the I3C bus, when the I3C peripheral acts as target.

Figure 680. RX-FIFO management, as target on the I3C bus



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed or private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

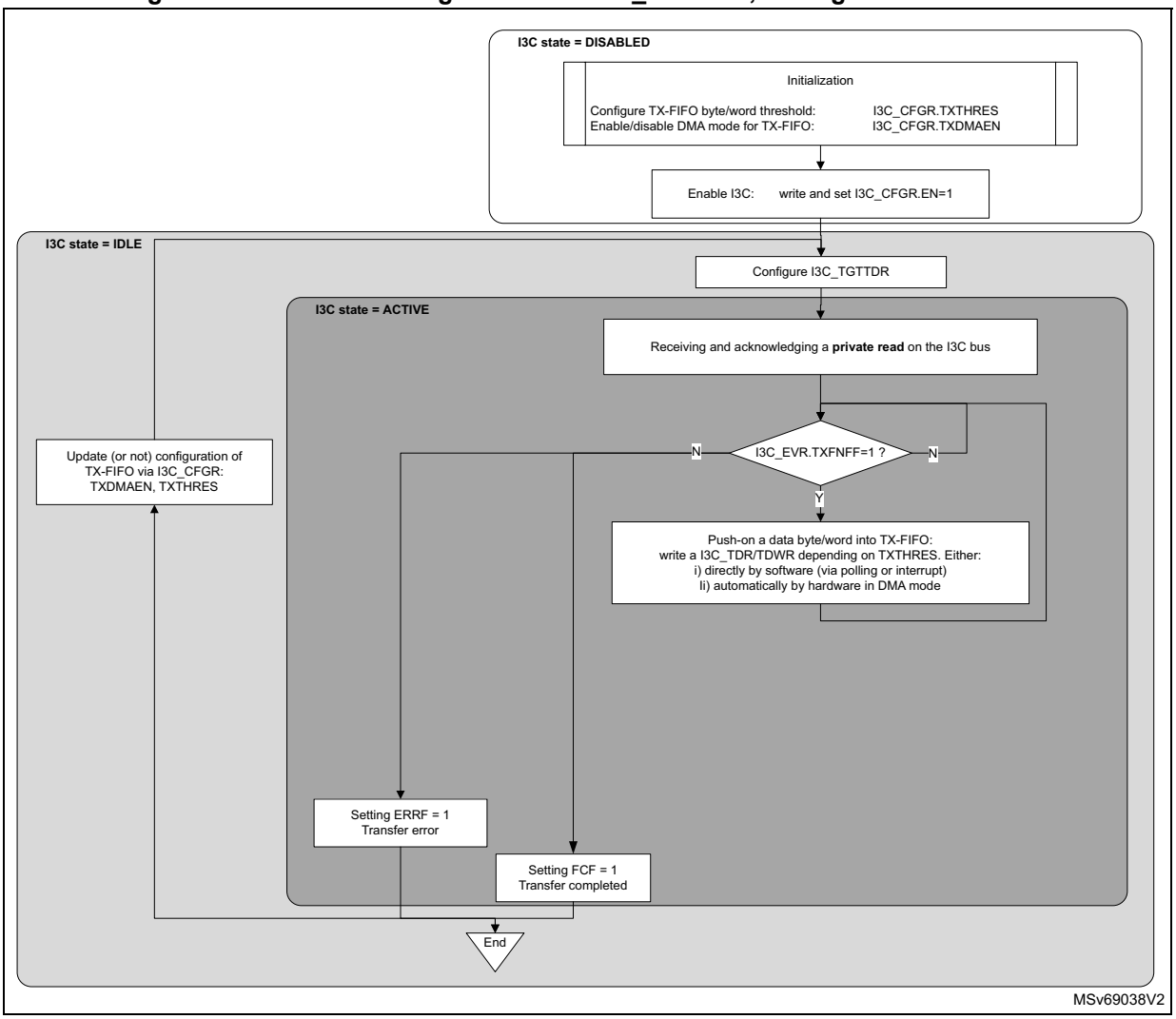
49.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 49.9*, the TX-FIFO is used only during a private read (*Figure 671*).

Figure 681 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.

Figure 681. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus , the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the serial bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 534).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the serial bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the serial bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the serial bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another read, depending upon the user application

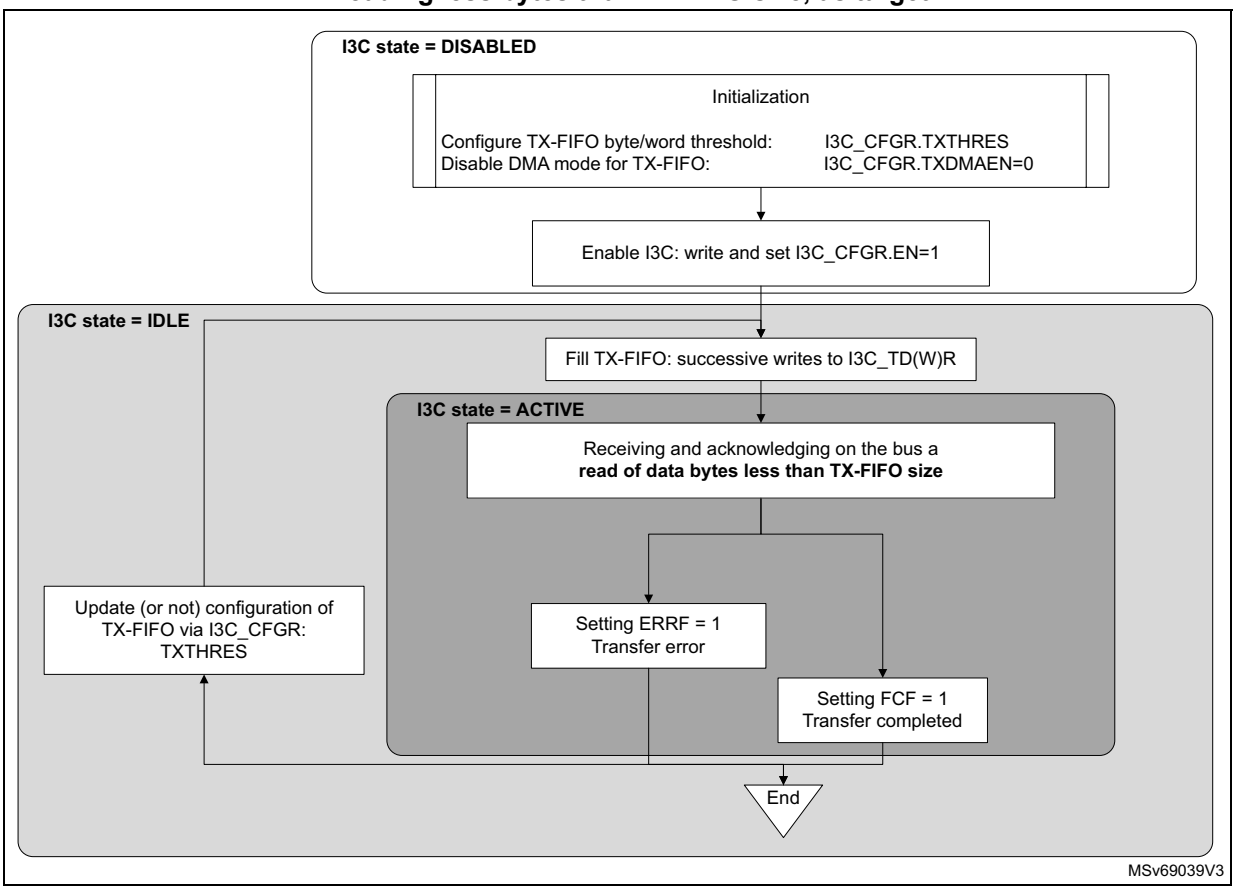
If TX-FIFO is empty and a data byte must be transmitted on the serial bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 682*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the serial bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 682. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



49.12 I3C error management

49.12.1 Controller error management

Table 543 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 543. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 543. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 543. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

49.12.2 Target error management

Table 544 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 544. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 544. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error of I3C bus	SDA is detected as driven at unexpected value on an data read (during a CCC direct read or a private read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C bus timings	Data to be transmitted not written by software or DMA in due time on a private read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

49.13 I3C wake-up from low-power mode(s)

49.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x = 1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 673*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 674*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 675*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.
7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a I3C reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start on the I3C bus

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the serial bus (SDA line is detected to be driven low while SCL is high)
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a bus start is missed and waits for the kernel clock to be provided. A controller message that is specifically addressed to this low-power device may have been missed (and NAcked). However the I3C peripheral can also detect a missed start if the controller started a CCC or private or legacy I^2C message to another addressed target, or if another target initiated a start for an IBI/CR/HJ request.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).
3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   - The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that a serial bus transaction was missed, but not whether the target was addressed or not. The software should use a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

49.14 I3C in low-power modes

Table 545. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Table 533* to know if Stop mode is supported, and which one.

49.15 I3C interrupts

Table 546. I3C interrupt requests

Interrupt acronym		Interrupt event	Used as		Interrupt enable: field in I3C_IER	Event flag: field in I3C_EVR	Event clear method: write 1 to field in I3C_CEVR
			Controller	Target			
I3C	EVT	A control word is requested	X	-	CFNFIE	CFNFF	-
		A status word is available	X	-	SFNEIE	SFNEF	-
		Data to be transmitted are requested	X	X	TXFNFIE	TXFNFF	-
		Received data are available	X	X	RXFNEIE	RXFNEF	-
		Controller: frame transfer is completed Target: private transfer is completed	X	X	FCIE	FCF	CFCF
		A private read transfer is prematurely ended by the target (and SMODE = 0 in the I3C_CFGR register)	X	-	RXTGTENDIE	RXTGTENDF	CRXTGTENDF
		An IBI request is received	X	-	IBIIE	IBIF	CIBIF
		A controller-role request is received	X	-	CRIE	CRF	CCRF
		A hot-join request is received	X	-	HJIE	HJF	CHJF
		IBI request is completed	-	X	IBIENDIE	IBIENDF	CIBIENDF
		Serial bus start is missed	-	X	WKPIE	WKPF	CWKPF
		Direct GETACCR CCC is received	-	X	CRUPDIE	CRUPDF	CCRUPDF
		Direct GETSTATUS CCC is received	-	X	STAIE	STAF	CSTAF
		Any direct GETxxx CCC (except GETSTATUS) is received	-	X	GETIE	GETF	CGETF
		Dynamic address is updated (broadcast ENTDAA/RSTDAA or direct SETNEWDA is received)	-	X	DAUPDIE	DAUPDF	CDAUPDF
		Direct SETMWL CCC is received	-	X	MWLUPDIE	MWLUPDF	CMWLUPDF
		Direct SETMRL CCC is received	-	X	MRLUPDIE	MRLUPDF	CMRLUPDF
		A reset pattern is detected	-	X	RSTIE	RSTF	CRSTF
		Bus activity state is updated (direct/broadcast ENTASx CCC is received)	-	X	ASUPDIE	ASUPDF	CASUPDF
		Broadcast/direct ENEC/DISEC CCC is received	-	X	INTUPDIE	INTUPDF	CINTUPDF
		Broadcast DEFTGTS CCC is received	-	X	DEFIE	DEFF	CDEFF
		Broadcast DEFGRPA CCC is received	-	X	GRPIE	GRPF	CGRPF
	ERR	An error occurred	X	X	ERRIE	ERRF	CERRF

49.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

49.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 49.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

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

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)
Condition: when I3C acts as I3C controller
0000: **SCL clock is forced to stop** until a next control word is executed
Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 543*.
0001: **header message**
Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 543* and MIPI specification about escalation handling.
0010: **private message** (refer to *Figure 670*)
Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:
– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).
0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 663*)
Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P
0100: **legacy I^2C message** (refer to *Figure 672*)
Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:
– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).
Others: reserved
Condition: when I3C acts as I3C target
1000: **hot-join request** (W) (refer to *Figure 674*)
The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
1001: **controller-role request** (W) (refer to *Figure 675*)
The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 673*)
Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.
Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.
0: write message
1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.
When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).
Linear encoding up to 64 Kbytes - 1
0x0000: no data to transfer
0x0001: 1 byte
0x0002: 2 bytes
...
0xFFFF: 64 Kbytes - 1 byte

49.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

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

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 542*, *Figure 663*, *Figure 664*, *Figure 665*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr

– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.

– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 664*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P

– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.

– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 664).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 665).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

49.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w									w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 49.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).

- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.

- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)

49.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

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

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

49.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received

word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.

- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

49.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non

full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.

– when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

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

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

49.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0]

in the I3C_TGTDR register.

- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

49.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

49.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

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

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

49.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
r	r	r	r	r	r	r	r						r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

49.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

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

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

49.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

49.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

49.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

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

49.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
w	w	w	w	w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

49.16.16 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)
This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.
This bit is cleared by hardware when the target receives a frame start.
When the device is not in Stop mode:
– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.
When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset:

– the software writes and sets the SYSRESETREQ control bit of the AIRCR register, when the device is controlled by a Cortex®-M.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: IBI request disabled

1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

49.16.17 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

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

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed

1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.

- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;

- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

49.16.18 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

49.16.19 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

49.16.20 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:

$t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK}$

Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.

Note: With I^2C Fm+ device t_{DIG_Hmin} = 260 ns, with I^2C Fm+ device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

$t_{SCLL_OD} = (SCLL_OD + 1) \times t_{I3CCLK}$

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG_OD_L}$(I3C) timings.

Note: With I^2C Fm+ device t_{DIG_Lmin} = 500 ns, with I^2C Fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG_OD_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

$t_{SCLH_I3C} = (SCLH_I3C + 1) \times t_{I3CCLK}$

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG_H_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG_H_MIXEDmin}$ = 32 ns and $t_{DIG_H_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

$t_{SCLL_PP} = (SCLL_PP + 1) \times t_{I3CCLK}$

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

49.16.21 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

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

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **SDA_HD**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD)] x t_{I3CCLK}

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C Fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C Fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: t_{CASr} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD)] x t_{I3CCLK}.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: t_{CBP} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: t_{CBSr} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].

This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

49.16.22 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL}as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data (and on the ACK/NACK phase of data byte of a legacy I^2C read)

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

49.16.23 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable
0: I3C target (no controller capable)
1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

49.16.24 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

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

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device

Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

49.16.25 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

49.16.26 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

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

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

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

Bits 2:0 Reserved, must be kept at reset value.

49.16.27 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]							
							rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

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

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

49.16.28 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

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

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID
This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).
This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector
This field is set as 0 (vendor fixed value).
This field represents bit[32] of the 48-bit provisioned ID.
Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID
This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.
This field represents bits[15:12] of the 48-bit provisioned ID.
Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

49.16.29 I3C register map

Table 547. I3C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0] / CCC[7:0]							RNW	DCNT[15:0]																
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	I3C_CFGR	Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0									0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0						0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							s_e_TP	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0

Table 547. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0									
0x05C	Reserved																																
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]								SCLL_OD[7:0]								SCLH_I3C[7:0]								SCLL_PP[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	I3C_TIMINGR1	Res.	Res.	Res.	SDA_HD	Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value				0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	Res.	Res.	Res.	STALLA	STALLC	STALLD	STALLT
	Reset value																	0	0	0	0	0	0	0	0					0	0	0	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0

Table 547. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR	Res.	Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS [1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

50 Universal synchronous/asynchronous receiver transmitter (USART/UART)

50.1 USART introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

50.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  – Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

50.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

50.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 548. Instance implementation

Instance	STM32H563/H573	STM32H523/H533
USART1	Full	Full
USART2	Full	Full
USART3	Full	Full
USART6	Full	Full
USART10	Full	-
USART11	Full	-
UART4	Basic	Basic
UART5	Basic	Basic
UART7	Basic	-
UART8	Basic	-
UART9	Basic	-
UART12	Basic	-
LPUART1	Low-power	Low-power

Table 549. USART/LPUART features

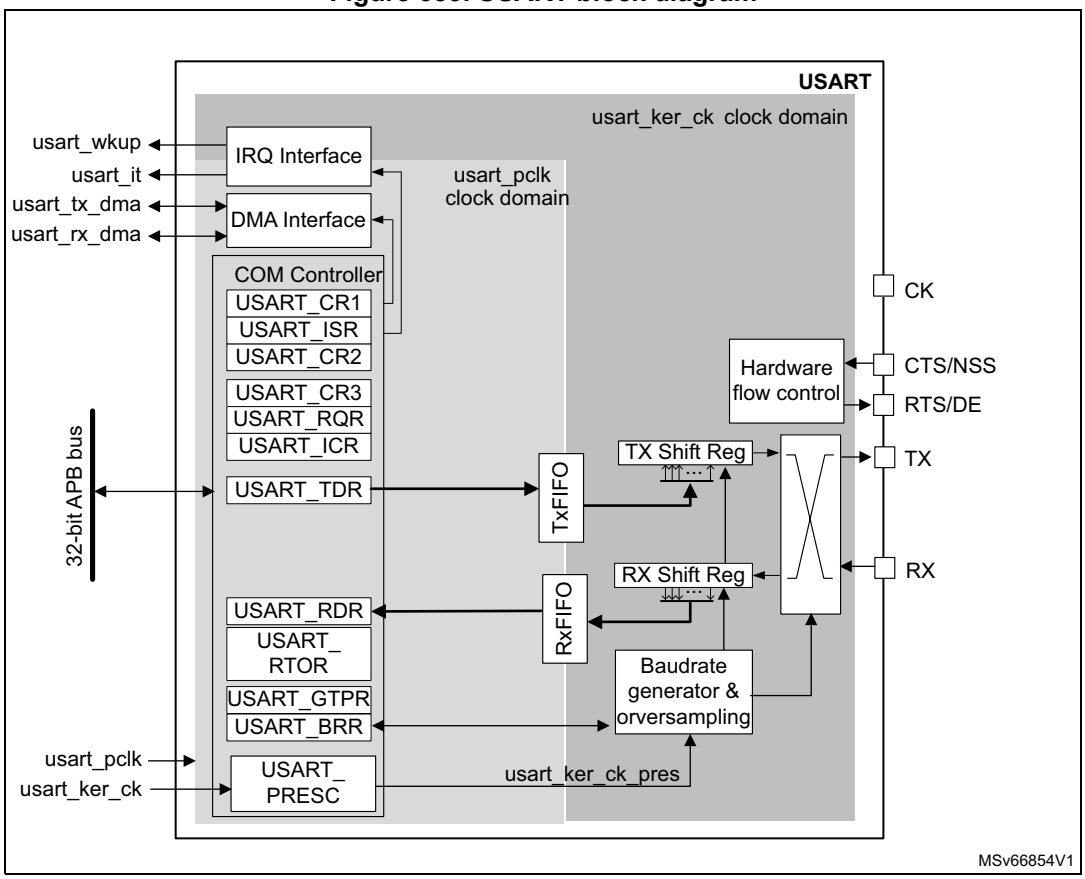
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop mode.

50.5 USART functional description

50.5.1 USART block diagram

Figure 683. USART block diagram



50.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 550* and *Table 551* for the list of USART input/output pins and internal signals.

Table 550. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 551. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request

50.5.3 USART clocks

The simplified block diagram given in *Section 50.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

50.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 684*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

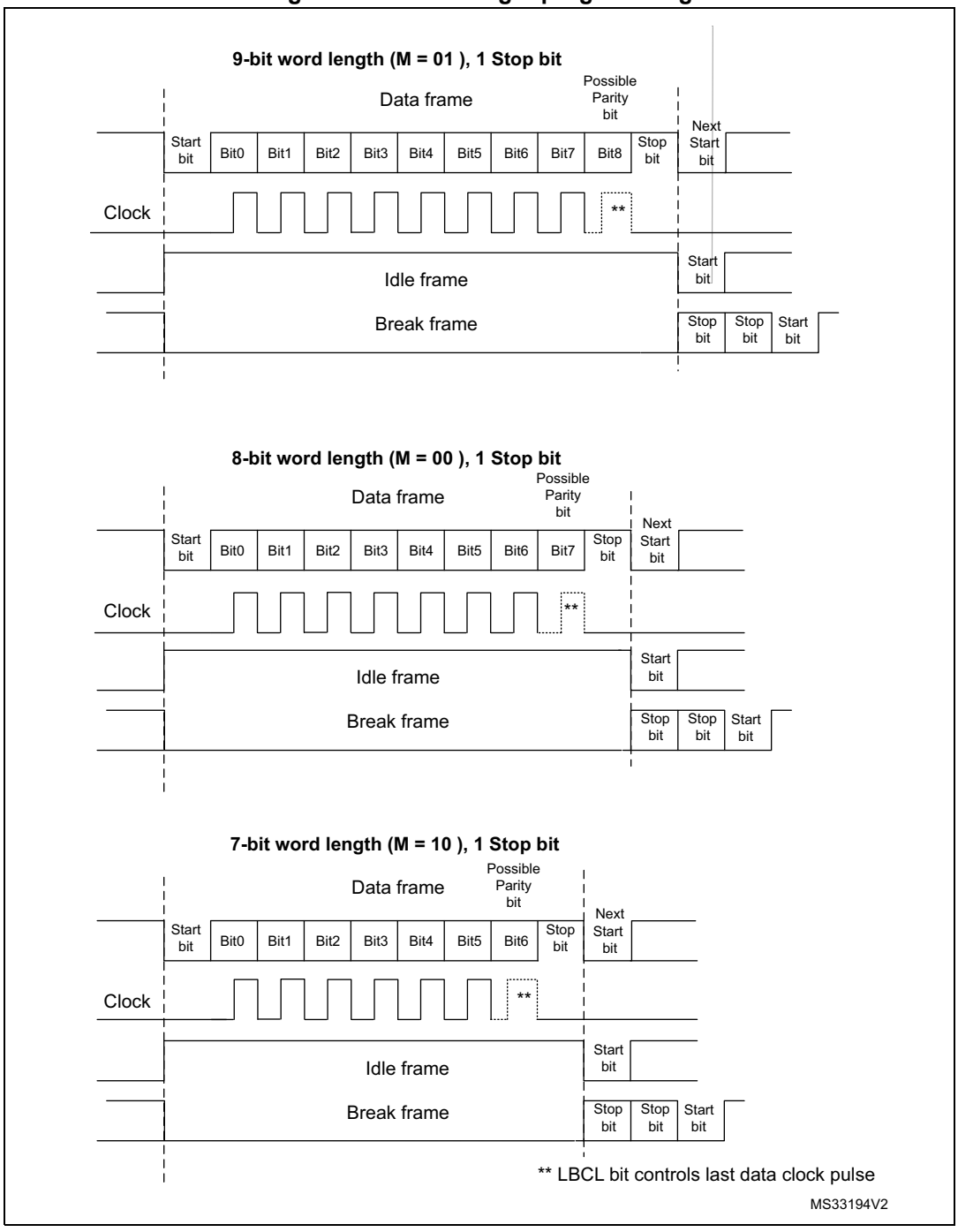
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 684. Word length programming



50.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

50.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

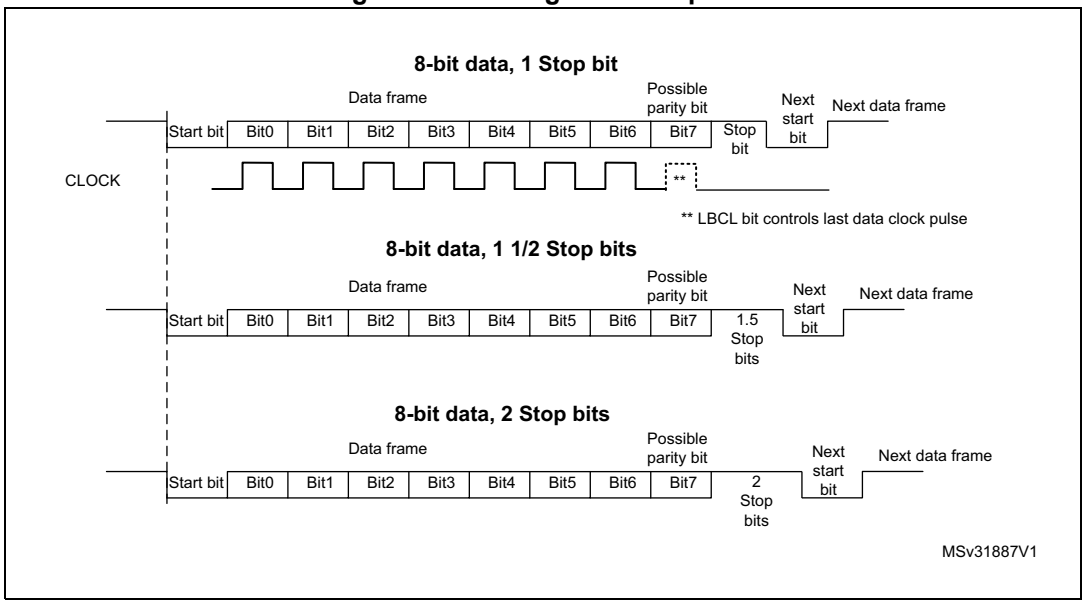
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 50.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 685. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 50.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

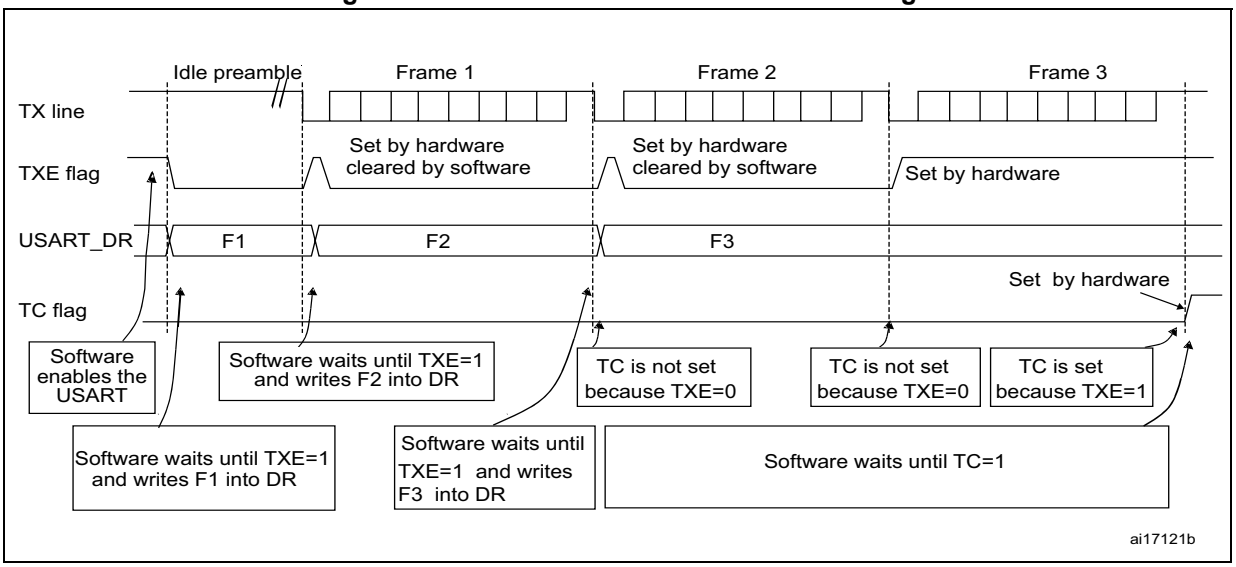
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 686*).

Figure 686. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 684*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

50.5.7 USART receiver

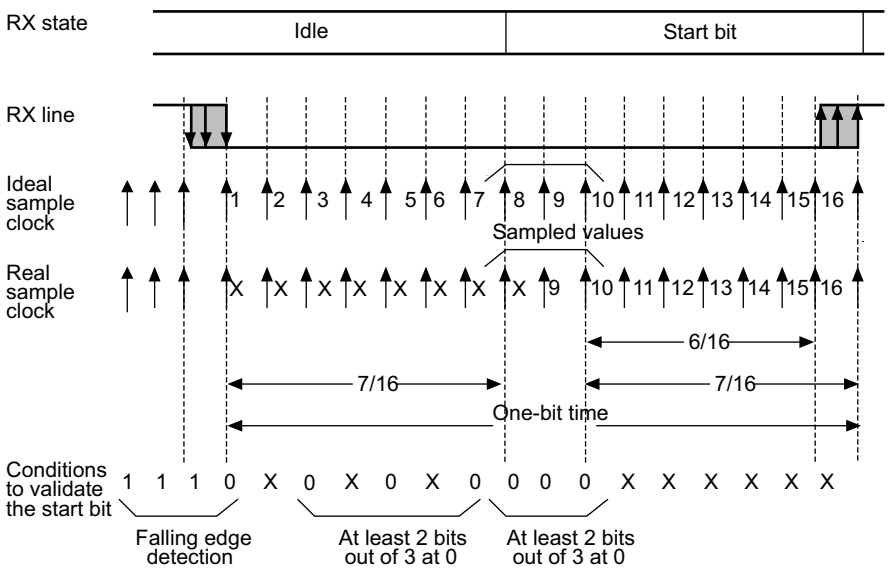
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 687. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 50.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

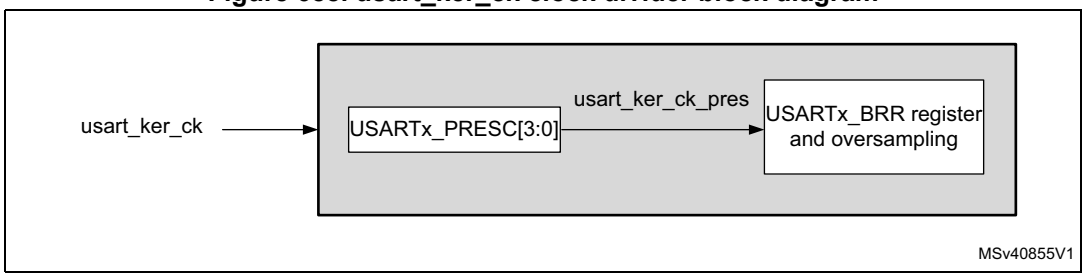
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 688. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 689* and *Figure 690*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 50.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 552*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 50.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 689. Data sampling when oversampling by 16

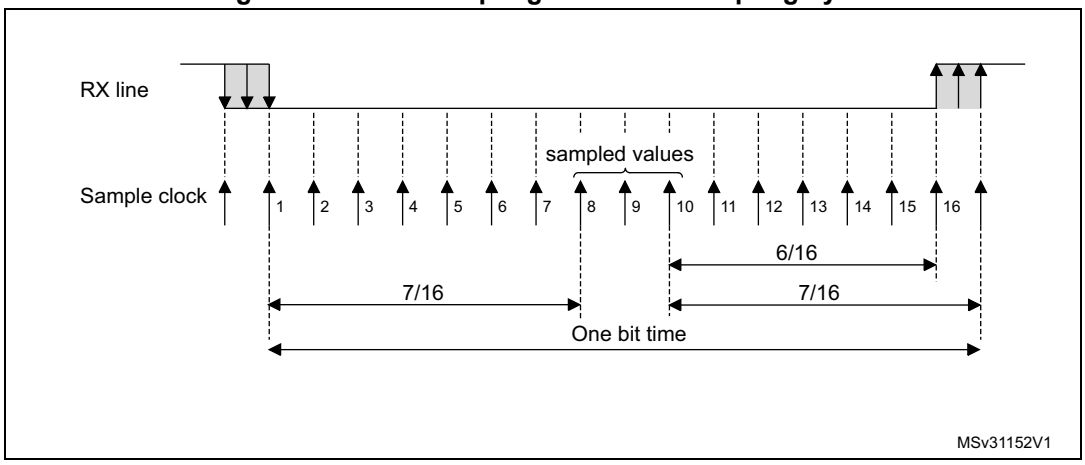


Figure 690. Data sampling when oversampling by 8

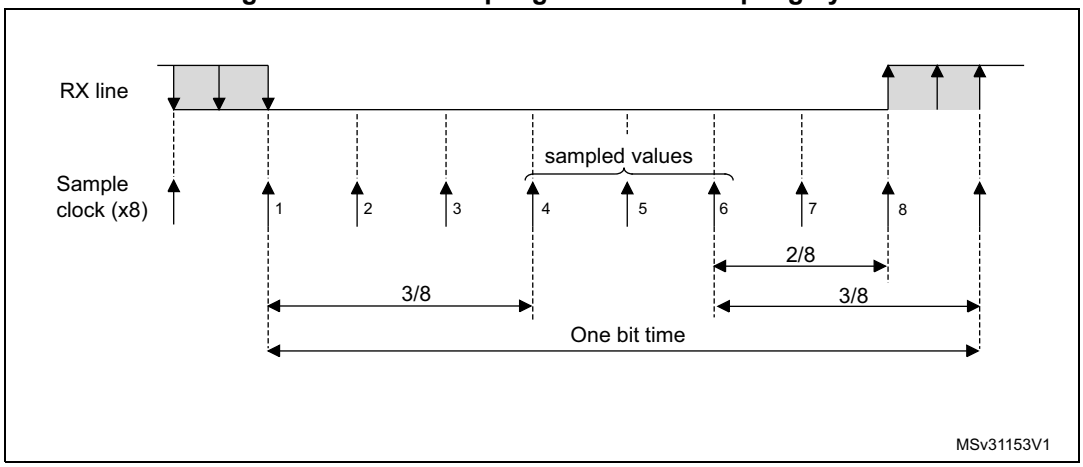


Table 552. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 50.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

50.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[3:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[3:0] = 0x681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[3:0] = USARTDIV = 52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[3:0] = 0x64

50.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 553*, *Table 554*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 553. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 554. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 553 and Table 554 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

50.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

50.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

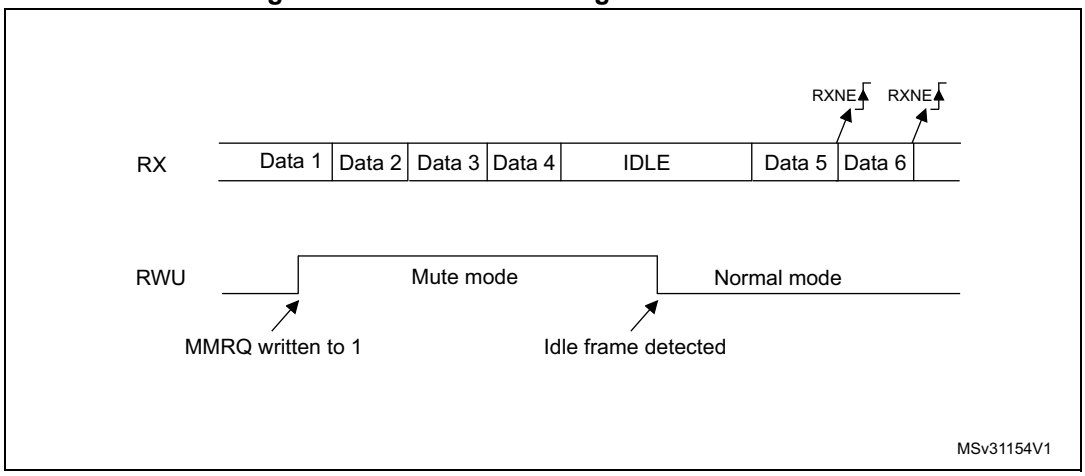
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 691*.

Figure 691. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

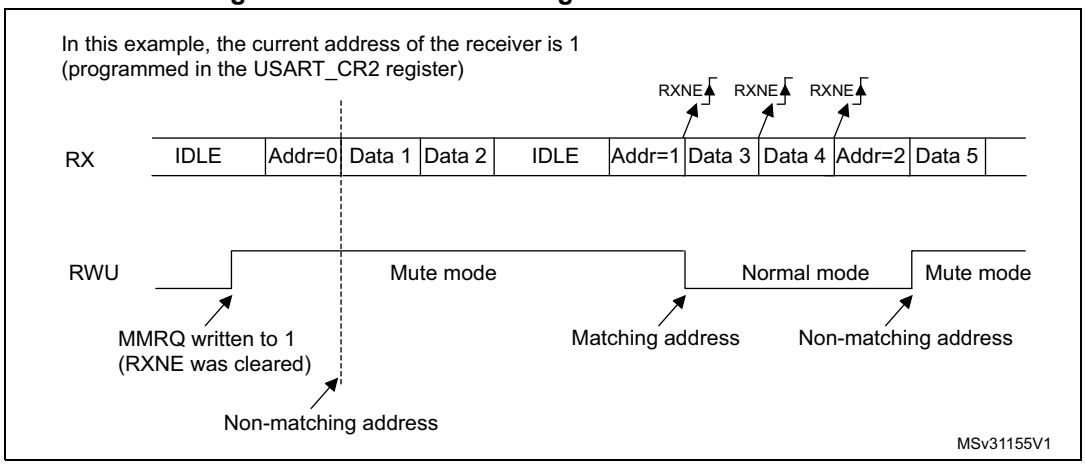
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 692*.

Figure 692. Mute mode using address mark detection



50.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

50.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 555*.

Table 555. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

50.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 50.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 50.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 693*.

Examples of break frames are given in *Figure 694*.

Figure 693. Break detection in LIN mode (11-bit break length - LBDL bit is set)

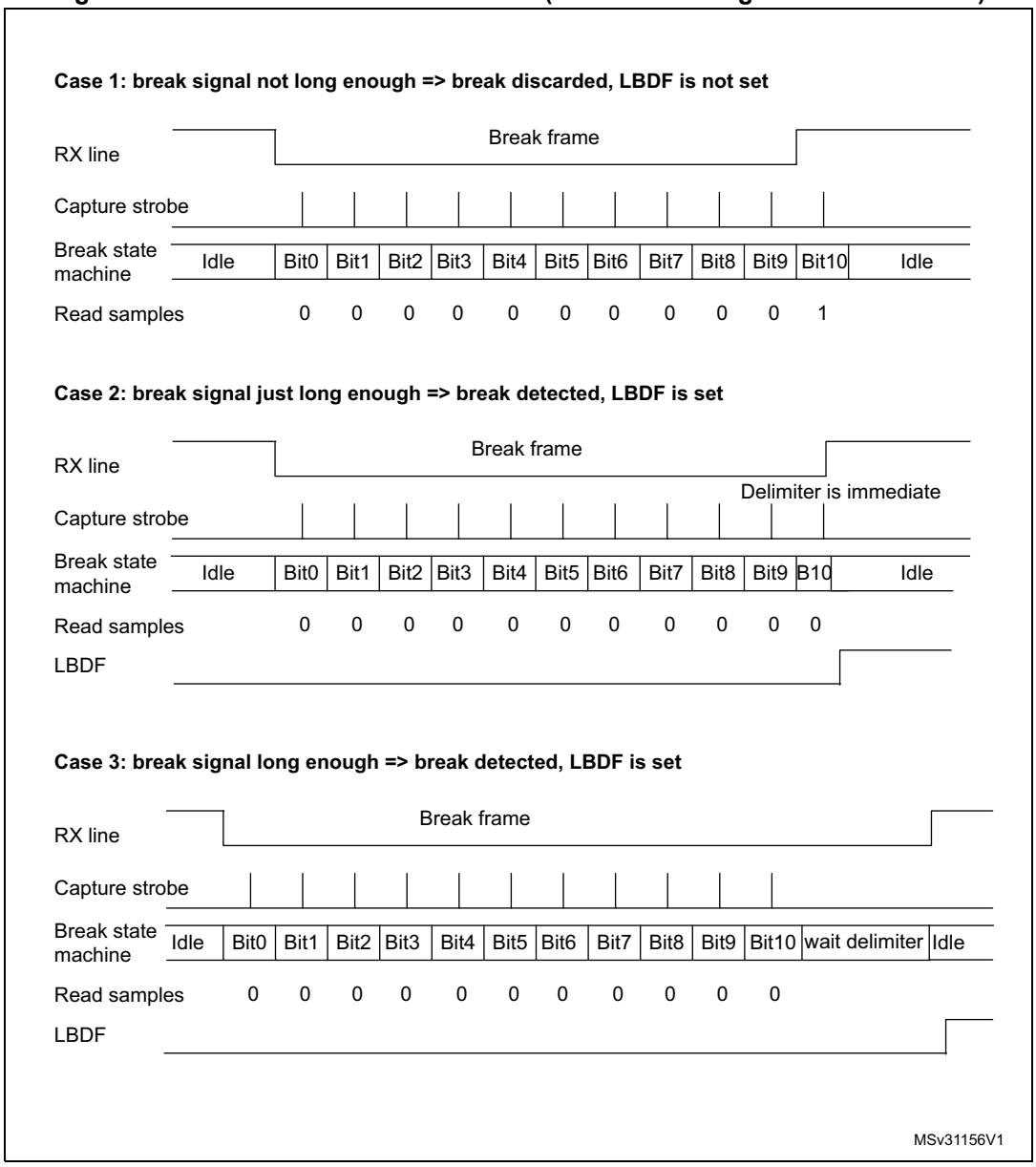
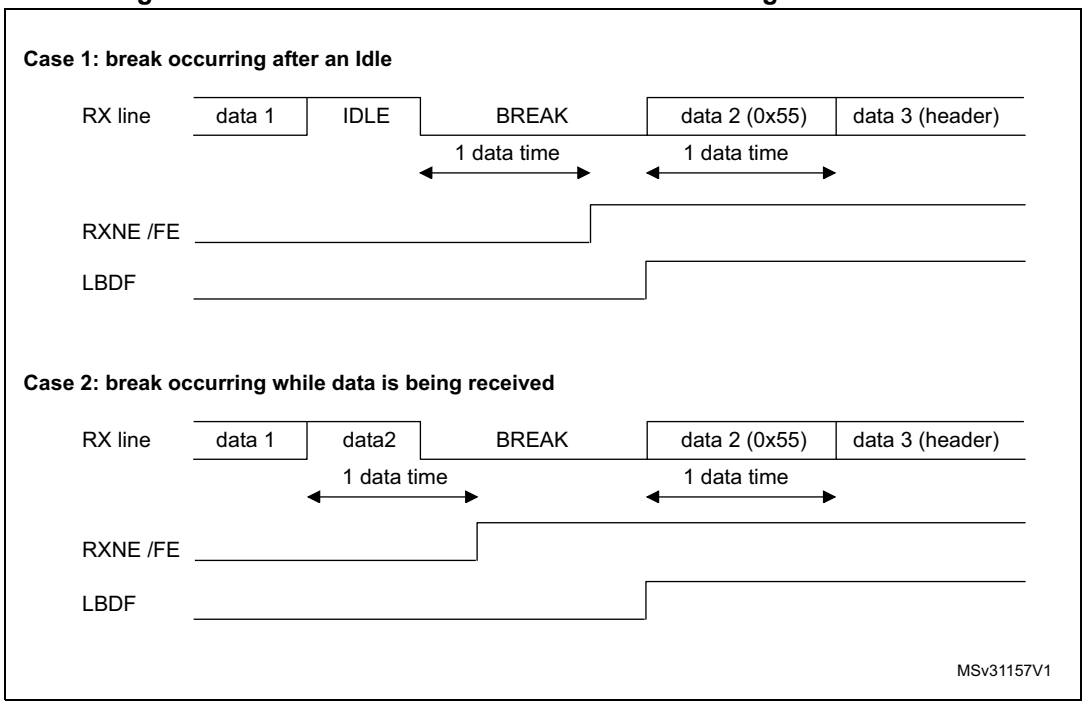


Figure 694. Break detection in LIN mode vs. Framing error detection



50.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 695*, *Figure 696* and *Figure 697*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 695. USART example of synchronous master transmission

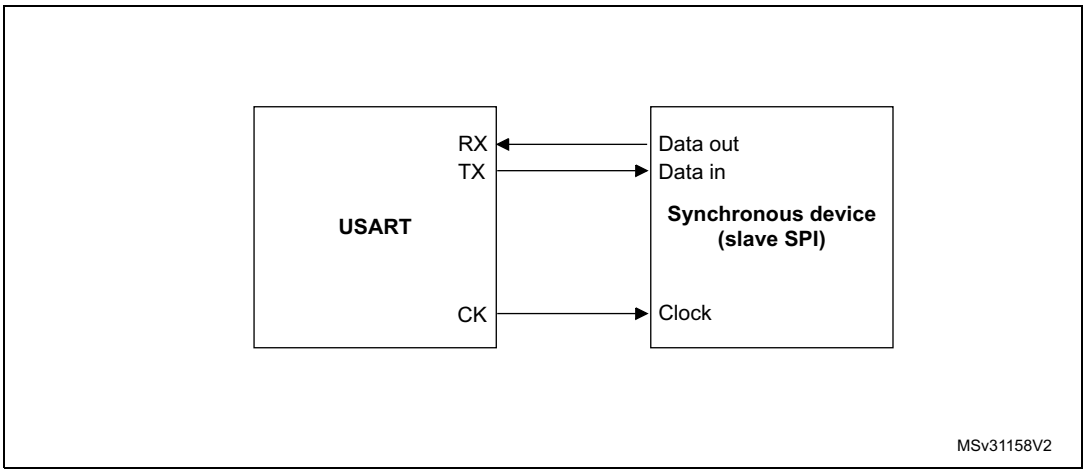


Figure 696. USART data clock timing diagram in synchronous master mode (M bits = 00)

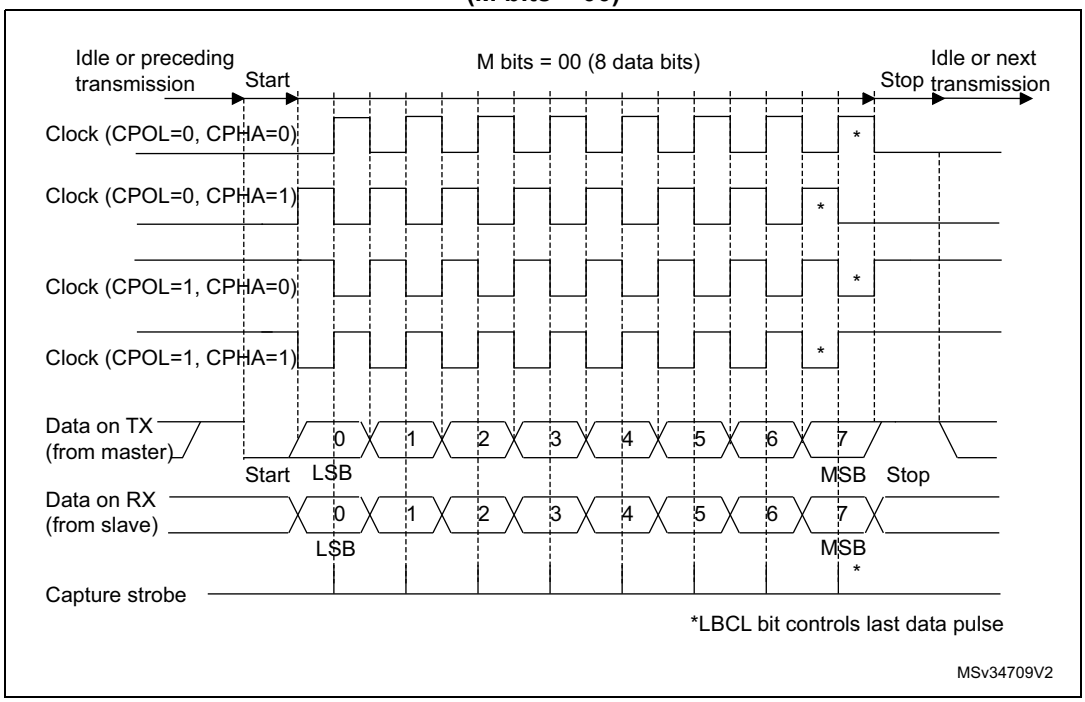
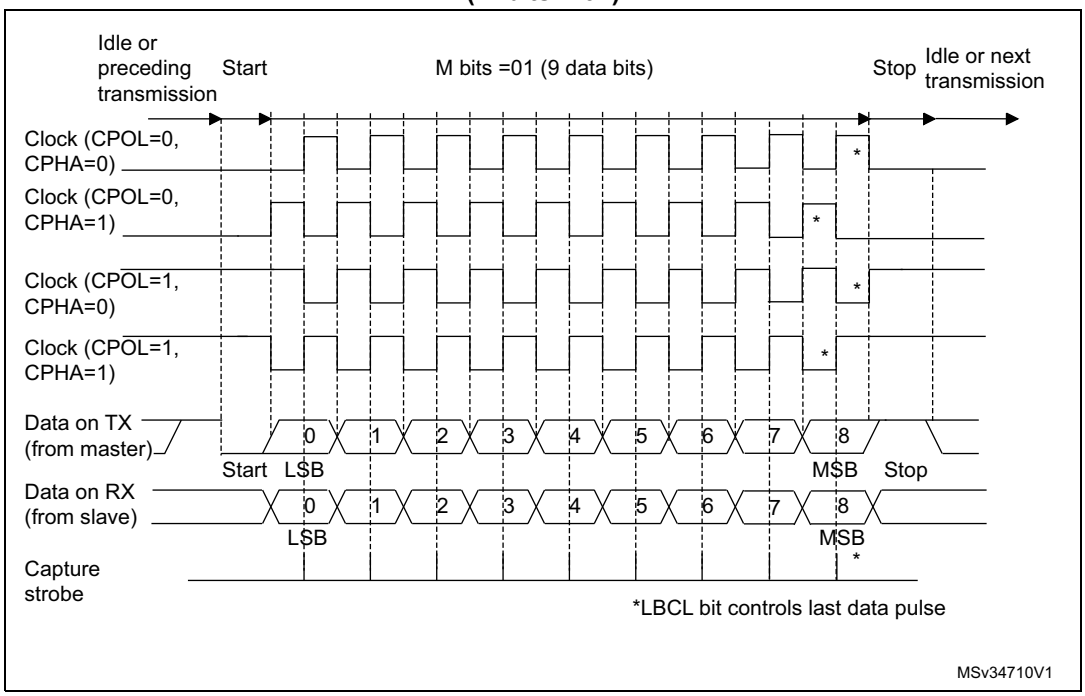


Figure 697. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

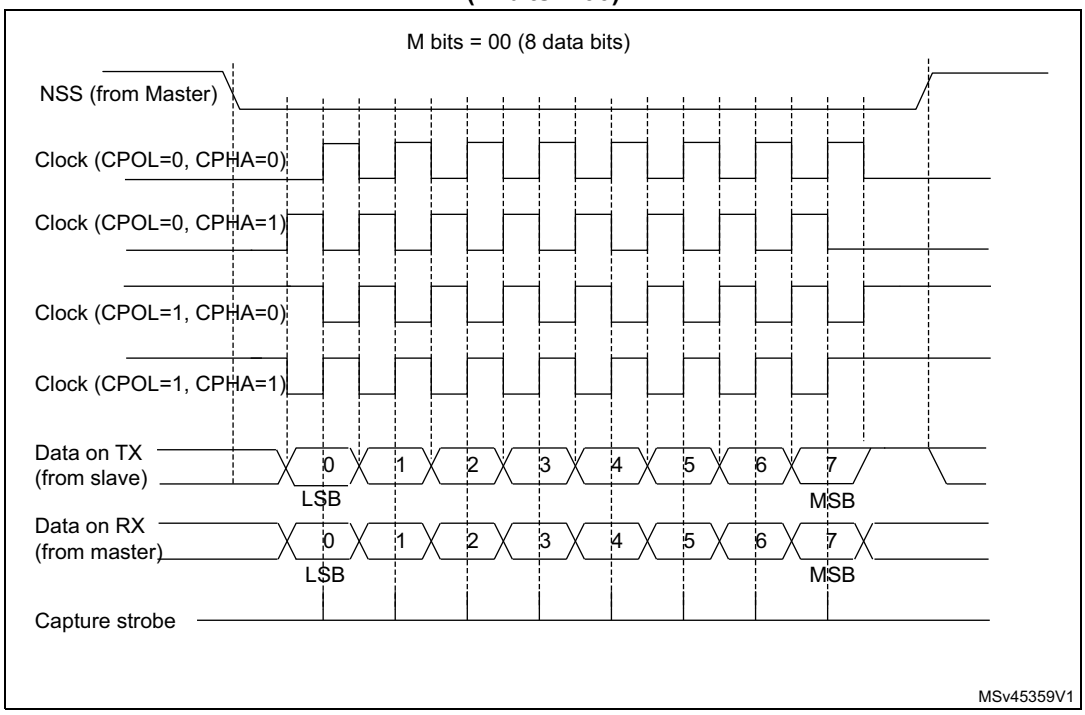
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 698*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 698. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

50.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

50.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

50.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 50.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

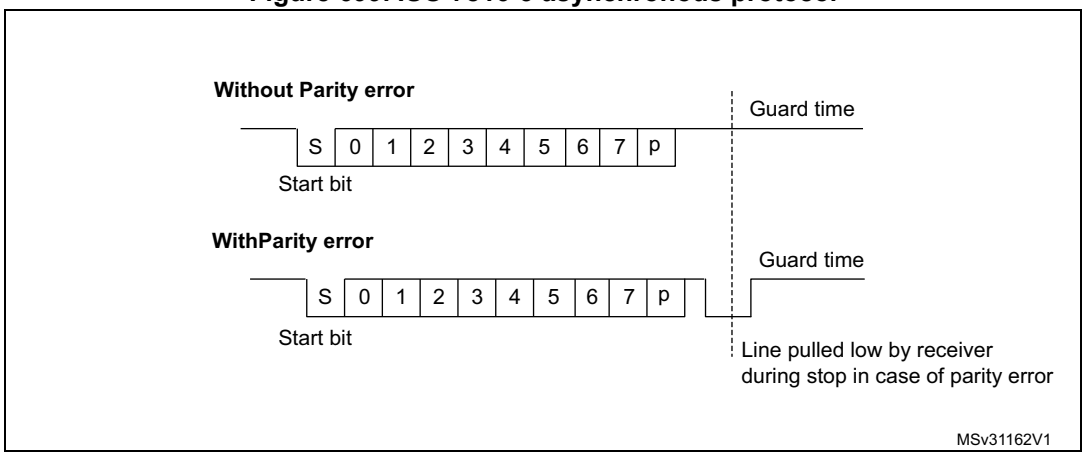
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 699 shows examples of what can be seen on the data line with and without parity error.

Figure 699. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

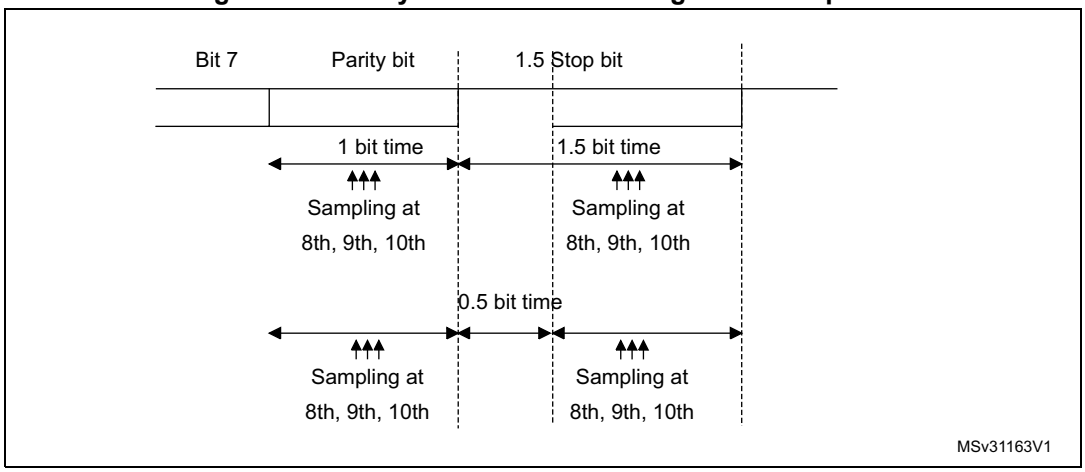
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 700 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 700. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

50.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 50.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 701*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 702*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 701. IrDA SIR ENDEC block diagram

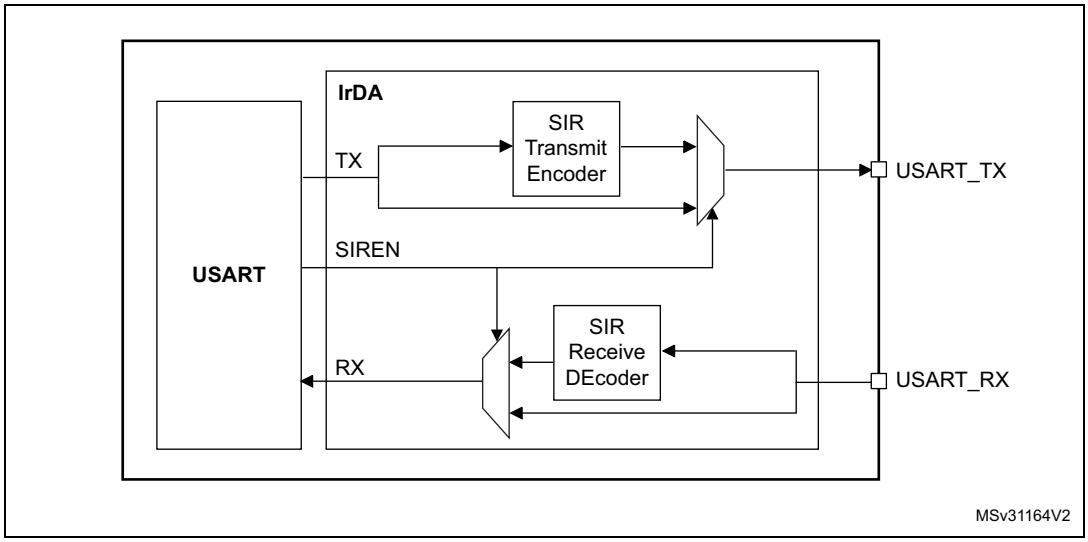
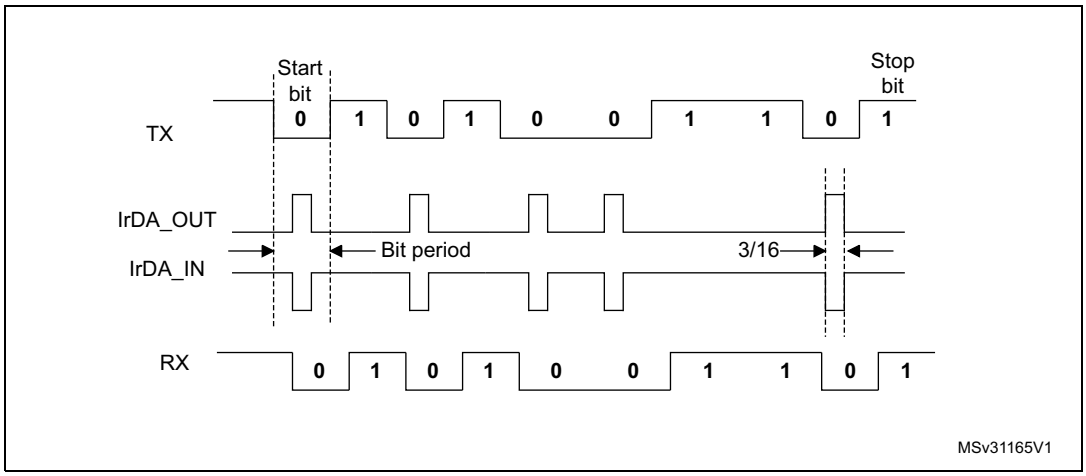


Figure 702. IrDA data modulation (3/16) - normal mode



50.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 50.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 50.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

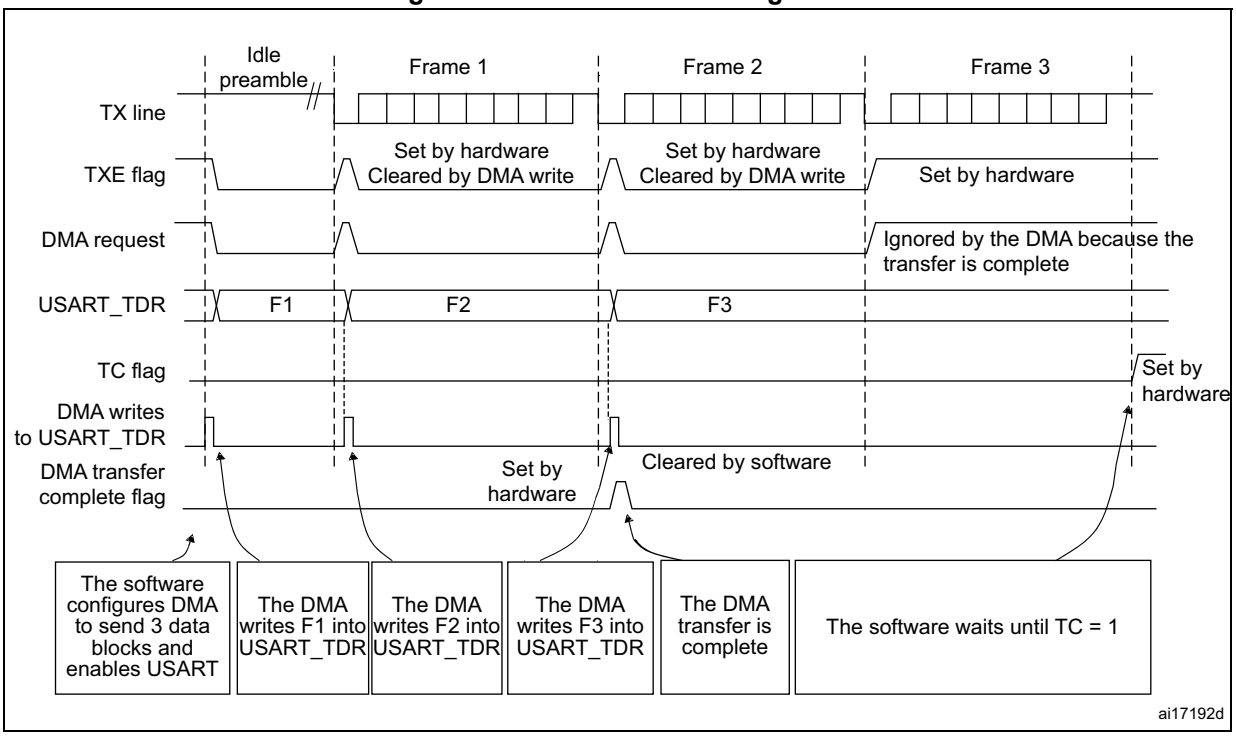
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 703. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

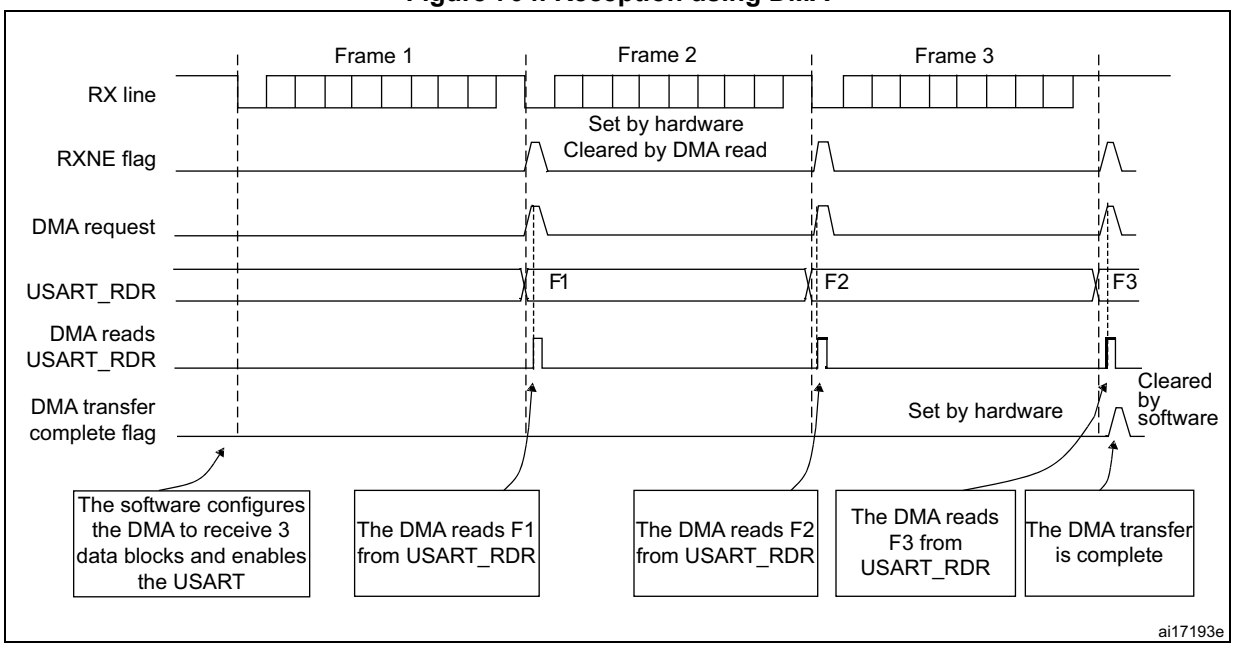
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 704. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

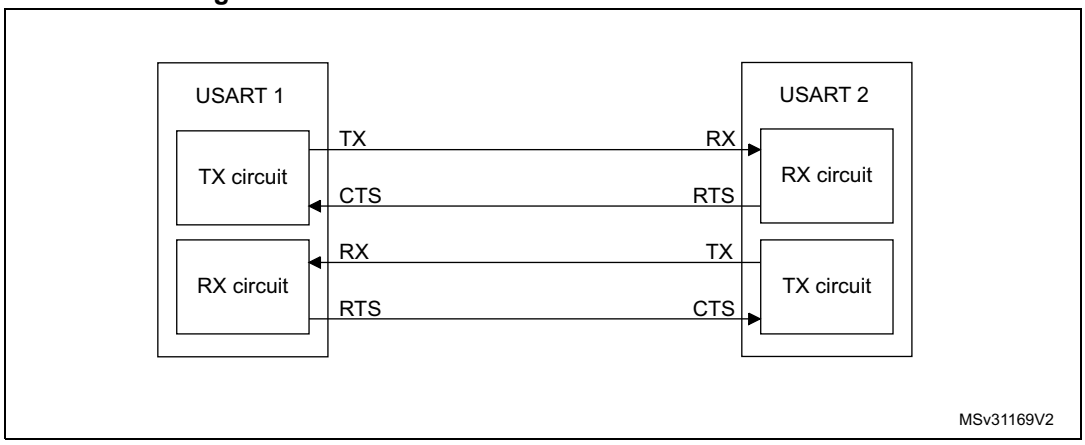
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

50.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 705* shows how to connect two devices in this mode:

Figure 705. Hardware flow control between two USARTs

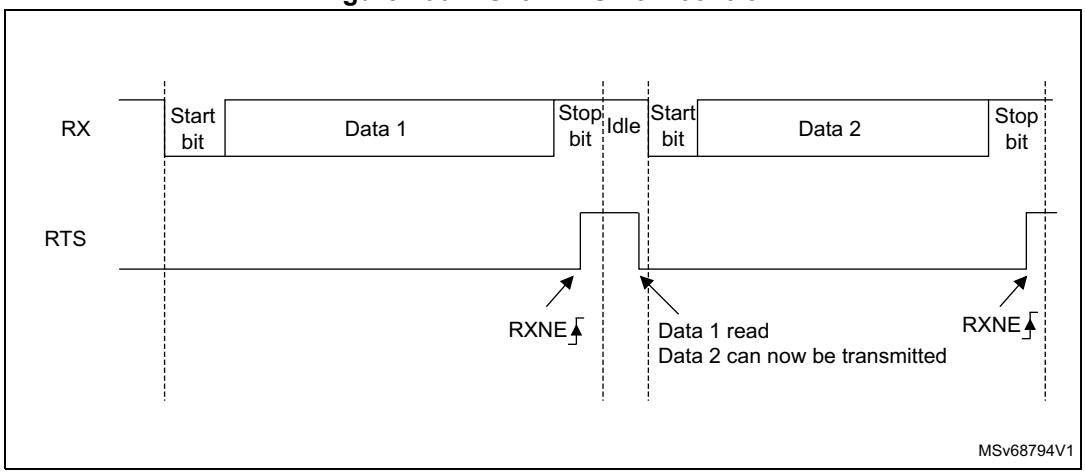


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 706* shows an example of communication with RTS flow control enabled.

Figure 706. RS232 RTS flow control



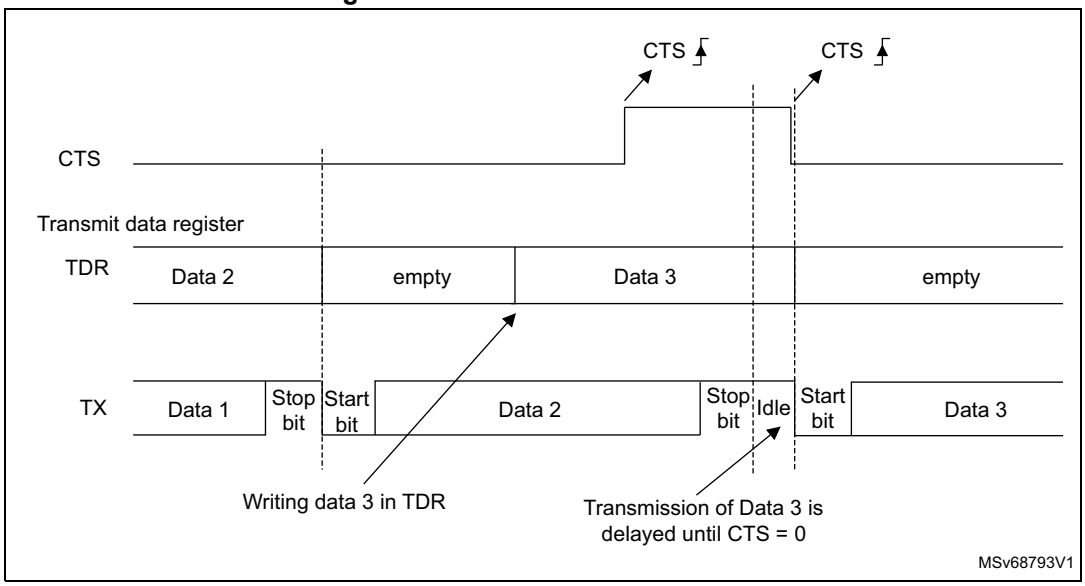
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 707* shows an example of communication with CTS flow control enabled.

Figure 707. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

50.5.22 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 708 shows the USART behavior when the wake-up event is verified.

Figure 708. Wake-up event verified (wake-up event = address match, FIFO disabled)

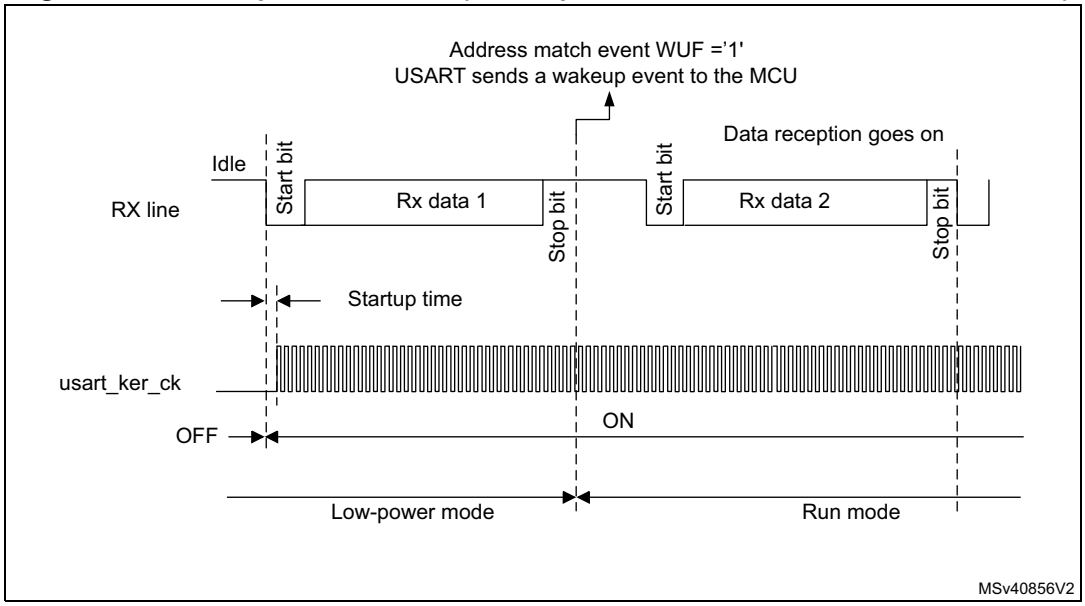
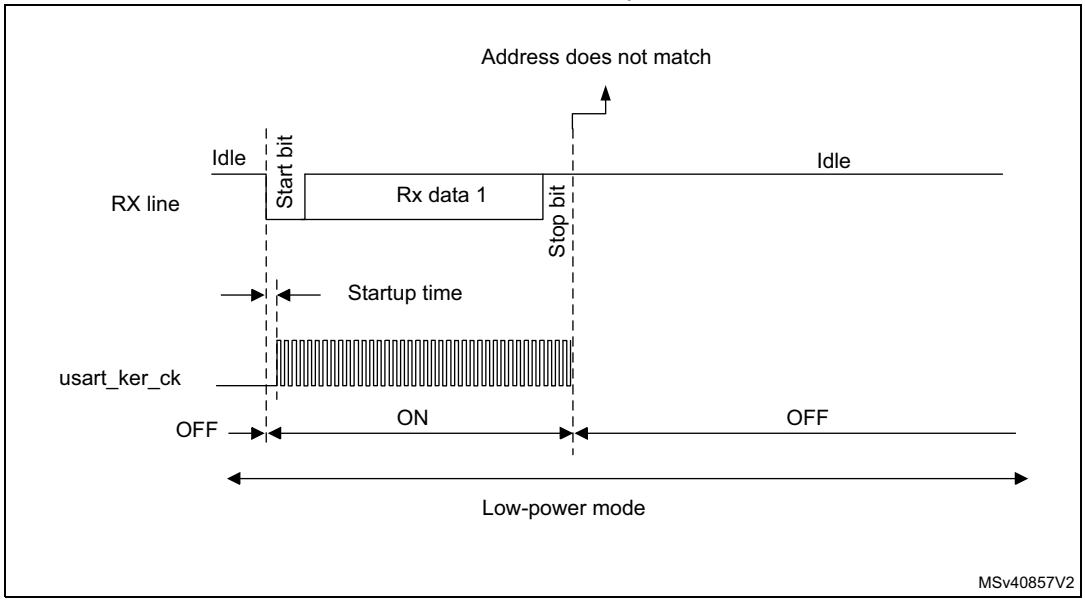


Figure 709 shows the USART behavior when the wake-up event is not verified.

Figure 709. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as a wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 50.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 553: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

50.6 USART in low-power modes

Table 556. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 50.4: USART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

50.7 USART interrupts

Refer to *Table 557* for a detailed description of all USART interrupt requests.

Table 557. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 557. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		No	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 50.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. When OVRDIS = 0.

50.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

50.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 50.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

50.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits

M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits

This bit can only be written when the USART is disabled (UE = 0).

Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 50.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

50.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 689* and *Figure 690*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

50.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2245.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2245.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation on page 2245.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 50.4: USART implementation on page 2245.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

50.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 50.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable
This bit is used for enabling smartcard mode.
0: Smartcard mode disabled
1: Smartcard mode enabled
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable
0: NACK transmission in case of parity error is disabled
1: NACK transmission during parity error is enabled
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power
This bit is used for selecting between normal and low-power IrDA modes
0: Normal mode
1: Low-power mode
This bit can only be written when the USART is disabled (UE = 0).
Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable
This bit is set and cleared by software.
0: IrDA disabled
1: IrDA enabled
This bit can only be written when the USART is disabled (UE = 0).
Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).
0: Interrupt inhibited
1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

50.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

50.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 50.4: USART implementation.

50.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 50.4: USART implementation.

50.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

50.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 50.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed, and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by a write to the USART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 50.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

50.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 50.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 50.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 50.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

50.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

50.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 50.5.1: USART block diagram*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

50.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 50.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

50.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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

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

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 50.4: USART implementation.

50.8.16 USART register map

Table 558. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 558. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_ RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

51 Low-power universal asynchronous receiver transmitter (LPUART)

51.1 LPUART introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

51.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

51.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 559. Instance implementation on STM32H5

Instance	STM32H563/H573	STM32H523/H533
USART1	Full	Full
USART2	Full	Full
USART3	Full	Full
USART6	Full	Full
USART10	Full	-
USART11	Full	-
UART4	Basic	Basic
UART5	Basic	Basic
UART7	Basic	-
UART8	Basic	-
UART12	Basic	-
LPUART1	Low-power	Low-power

Table 560. USART/LPUART features

Modes/features(1)	Full feature set	Basic feature set	Low-power feature set
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-

Table 560. USART/LPUART features (continued)

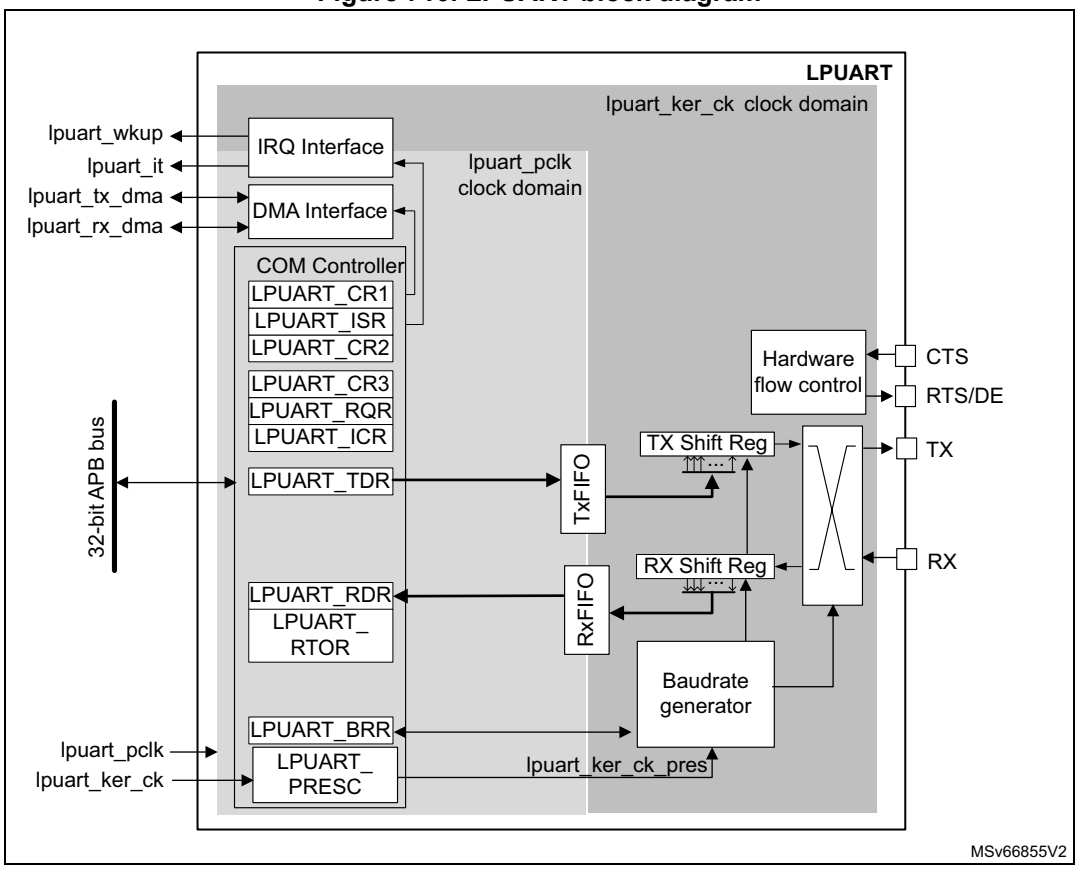
Modes/features[1]	Full feature set	Basic feature set	Low-power feature set
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	8		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. X = supported.
2. Wake-up supported from Stop mode.

51.4 LPUART functional description

51.4.1 LPUART block diagram

Figure 710. LPUART block diagram



51.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

    RX is the serial data input.

  – **TX (**Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pins:

  – **CTS (**Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request to send)

    When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 561* and *Table 562* for the list of LPUART input/output pins and internal signals.

Table 561. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 562. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 562. LPUART internal input/output signals (continued)

Signal name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request

51.4.3 LPUART clocks

The simplified block diagram given in *Section 51.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

51.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 684*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

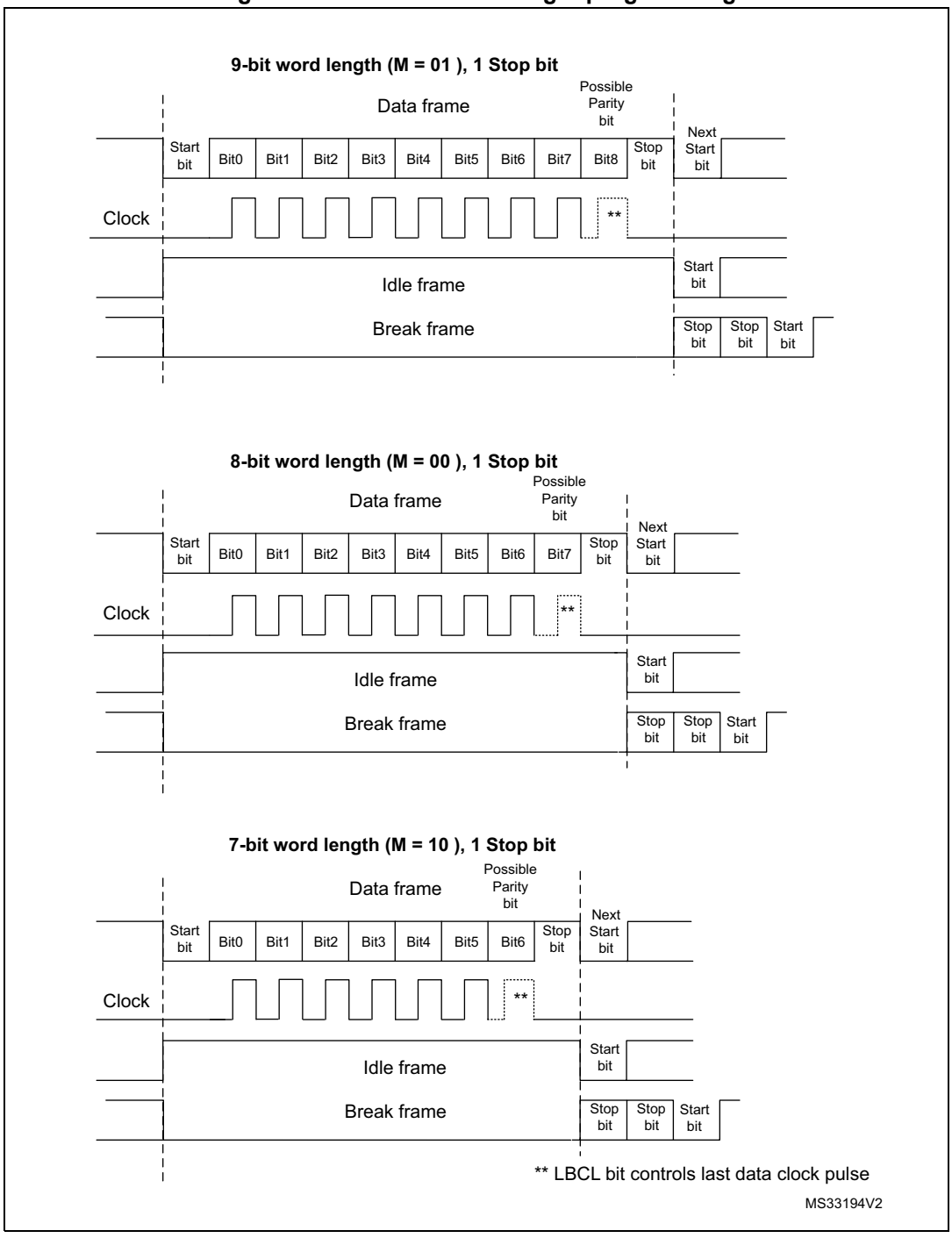
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 711. LPUART word length programming



51.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

51.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 51.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

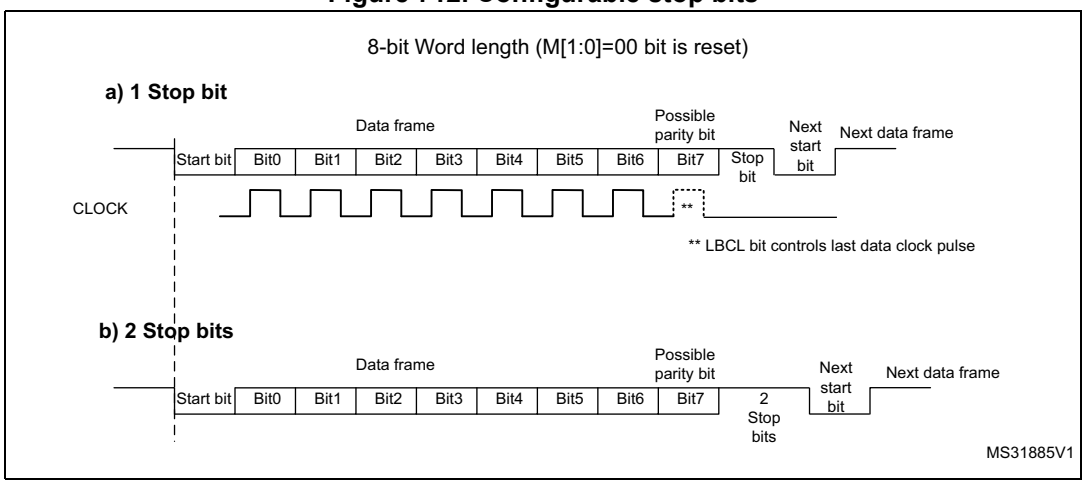
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 712. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 51.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

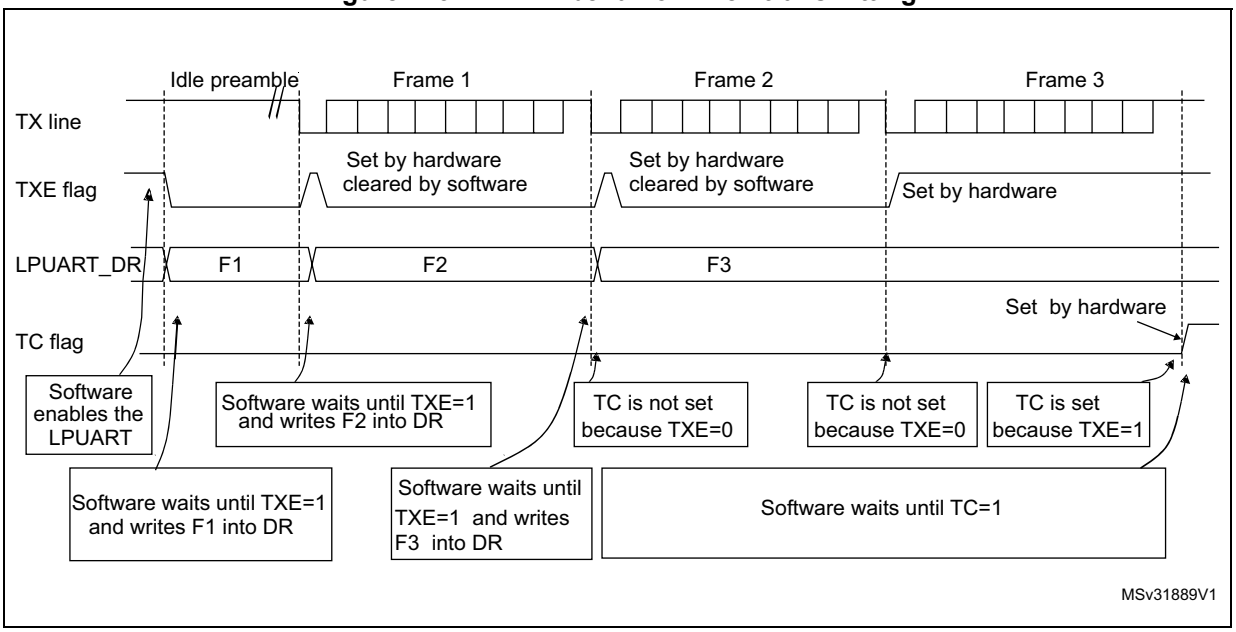
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 713: TC/TXE behavior when transmitting*).

Figure 713. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 711*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

51.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 51.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

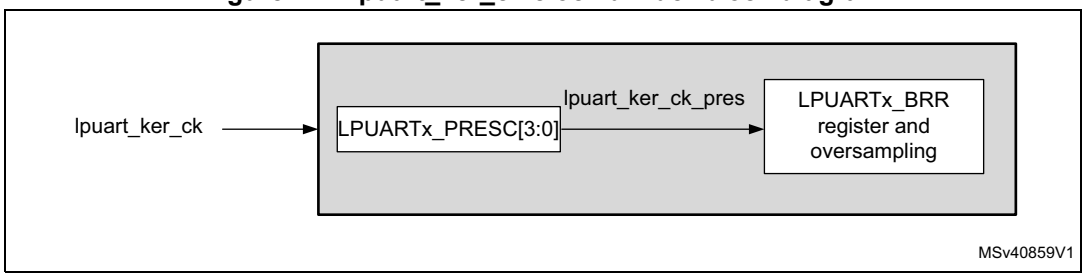
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 714. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

51.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 563. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	Desired	Actual	Value programmed in the baud rate register	% Error = (Calculated – Desired) B.rate / Desired B.rate
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

51.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 564*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 564. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 564 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

51.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

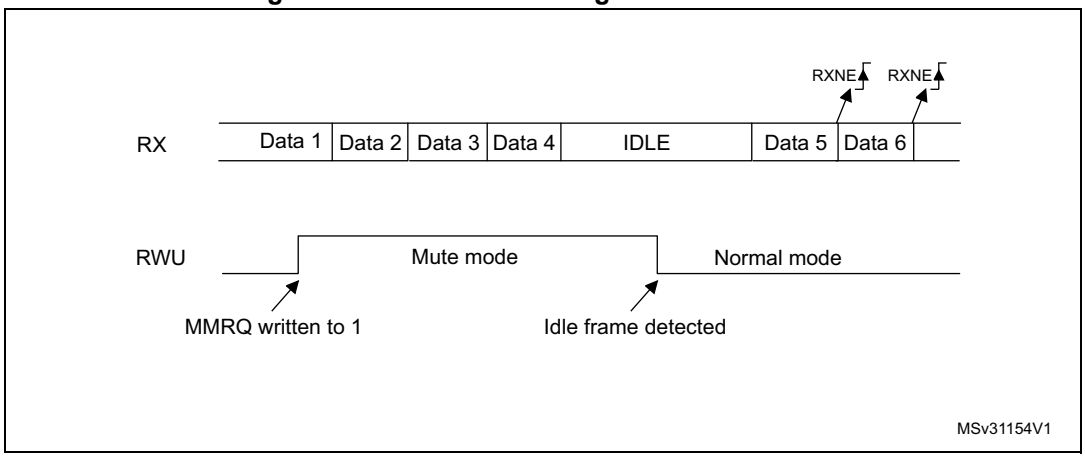
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 715*.

Figure 715. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

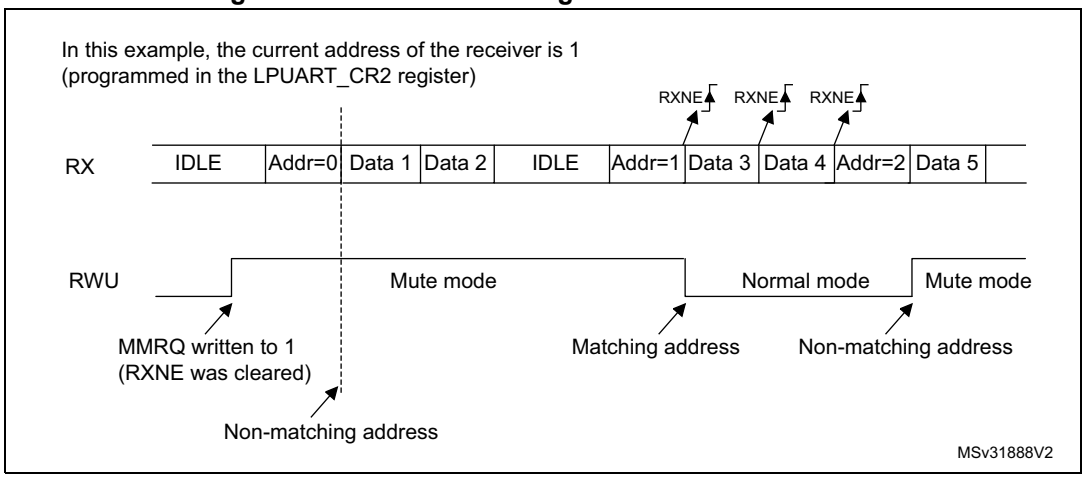
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 716*.

Figure 716. Mute mode using address mark detection



51.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 565*.

Table 565: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

51.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

51.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 51.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 51.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

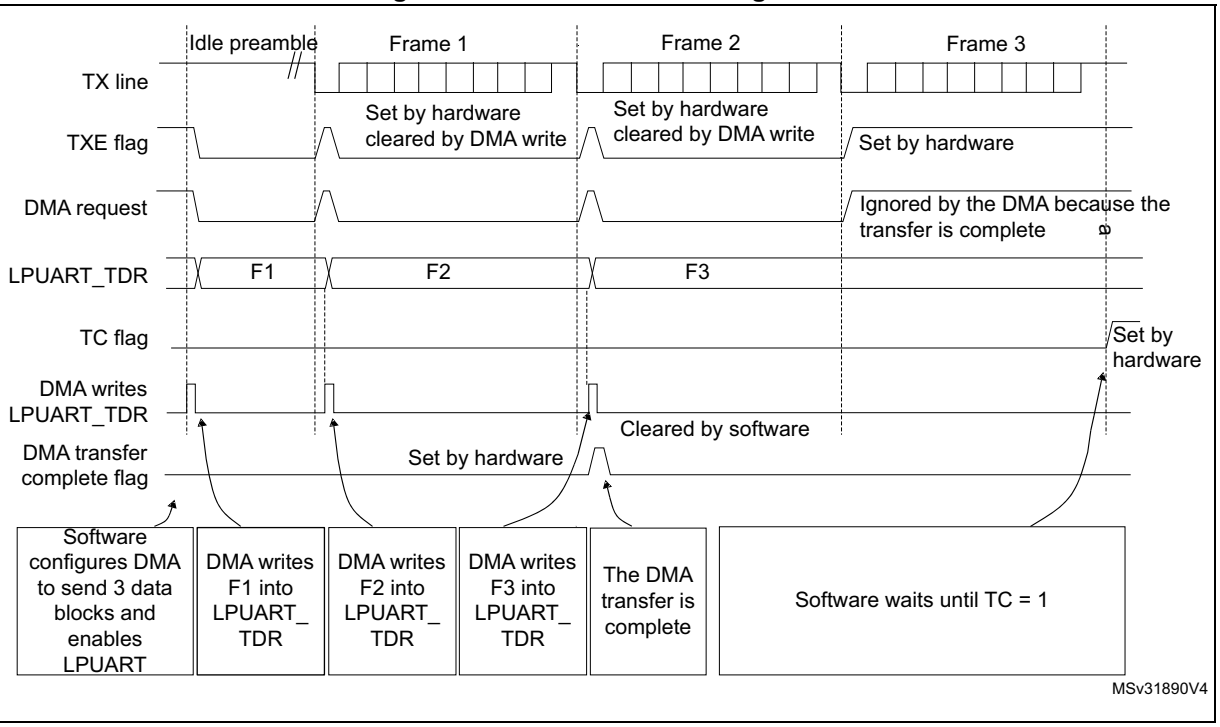
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 717. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

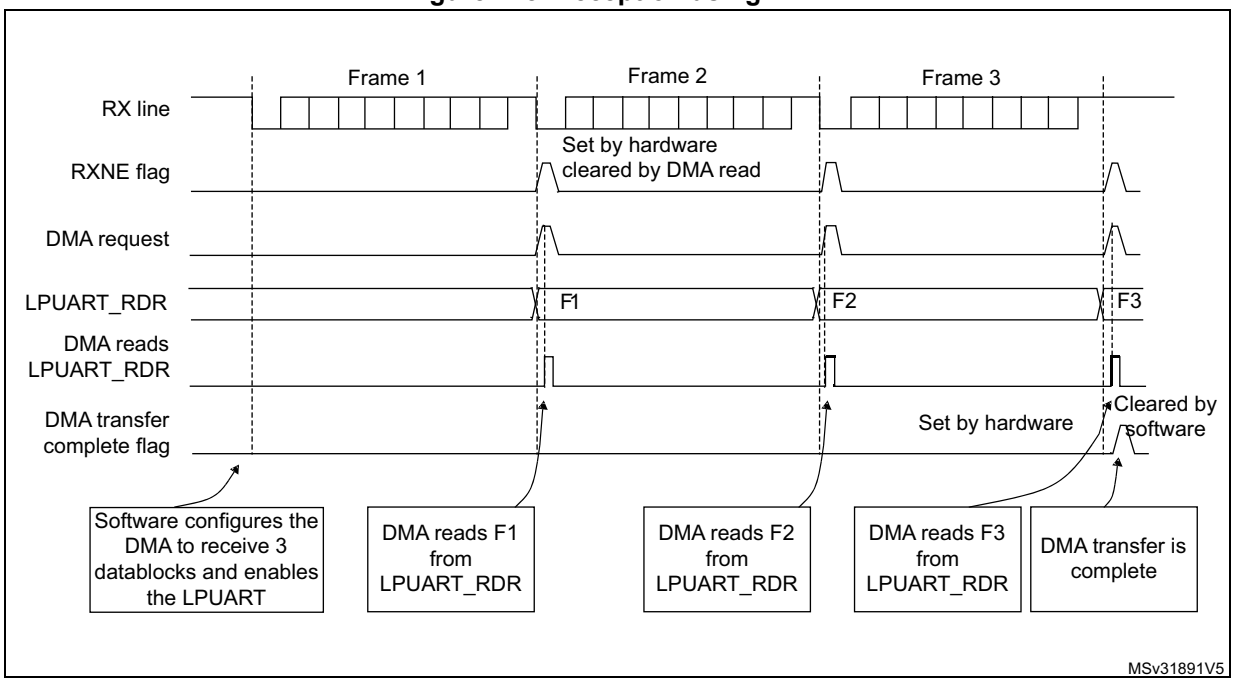
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 718. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

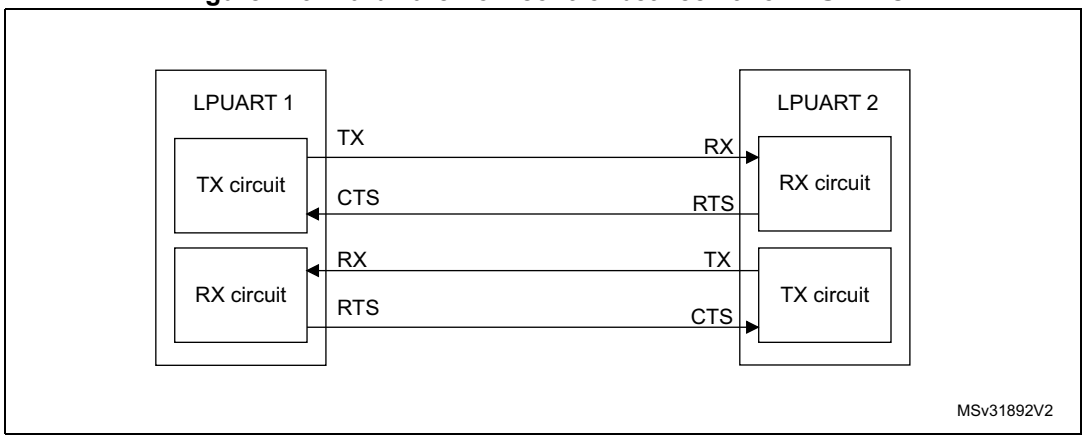
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

51.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 719* shows how to connect two devices in this mode.

Figure 719. Hardware flow control between two LPUARTs

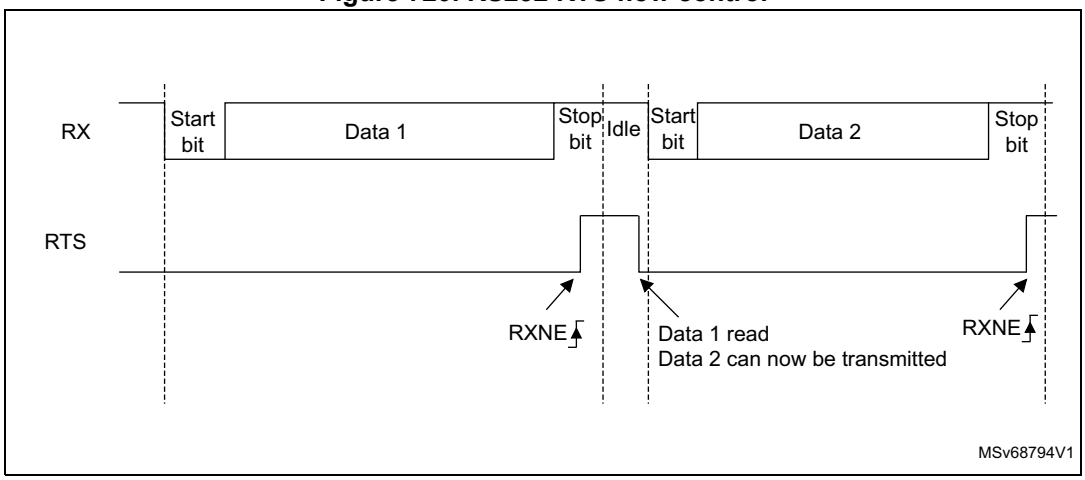


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 720* shows an example of communication with RTS flow control enabled.

Figure 720. RS232 RTS flow control



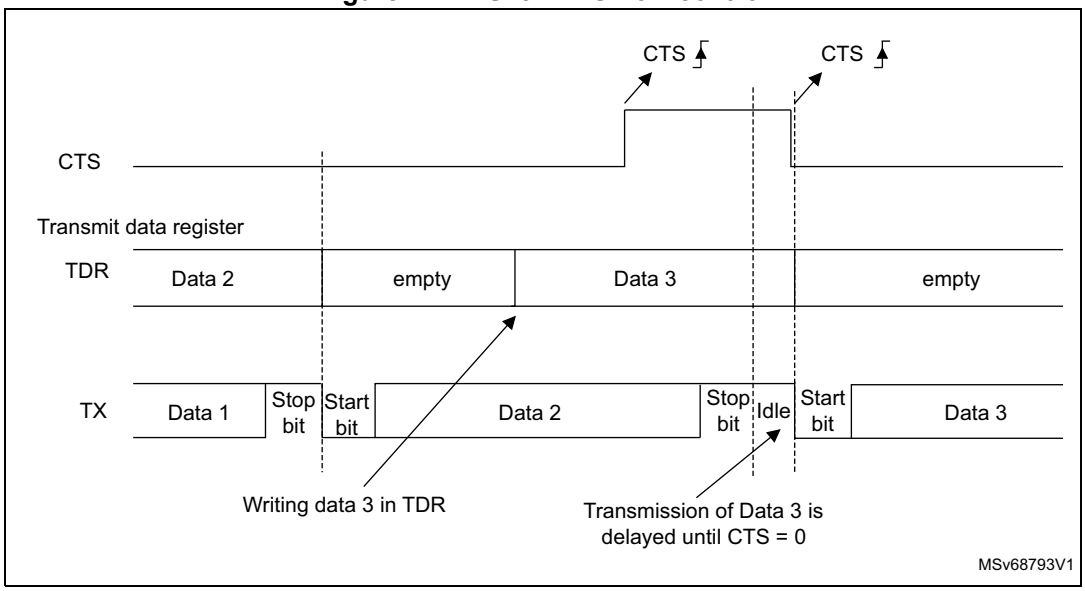
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 721* shows an example of communication with CTS flow control enabled.

Figure 721. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ($f_{lpuart_ker_ck_pres}$) cycles:

- The driver enable assertion time equals
  - (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  - (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

51.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the lpuart_pclk is gated, the LPUART provides a wake-up interrupt (**lpuart_wkup**) if a specific action requiring the activation of the **lpuart_pclk** clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many

LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **lpuart_wkup** interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and the **lpuart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure that the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not woken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 722 shows the behavior when the wake-up event is verified.

Figure 722. Wake-up event verified (wake-up event = address match, FIFO disabled)

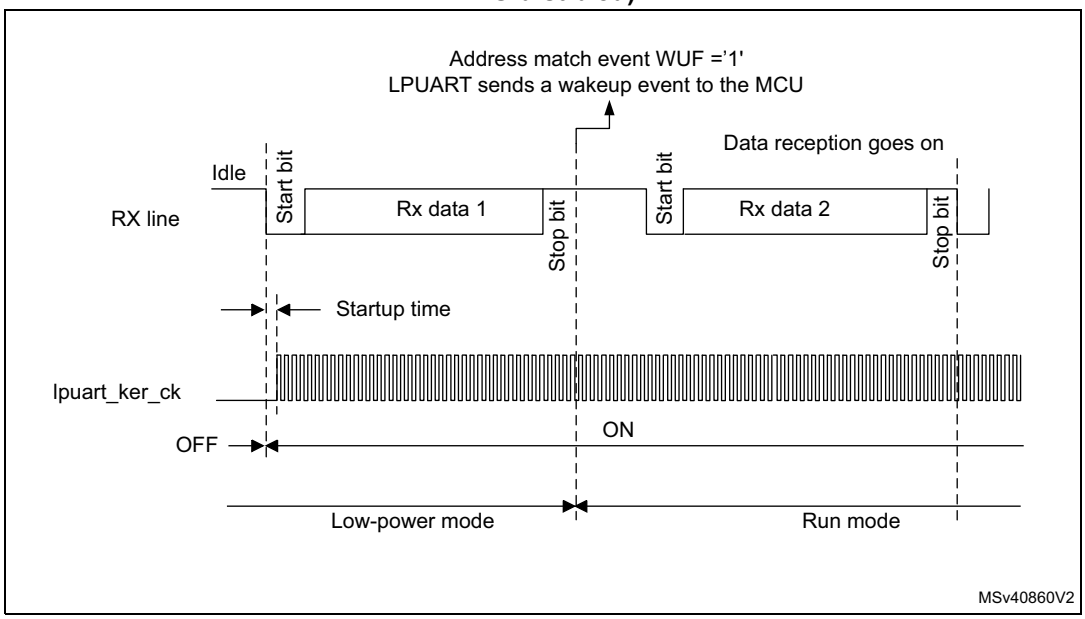
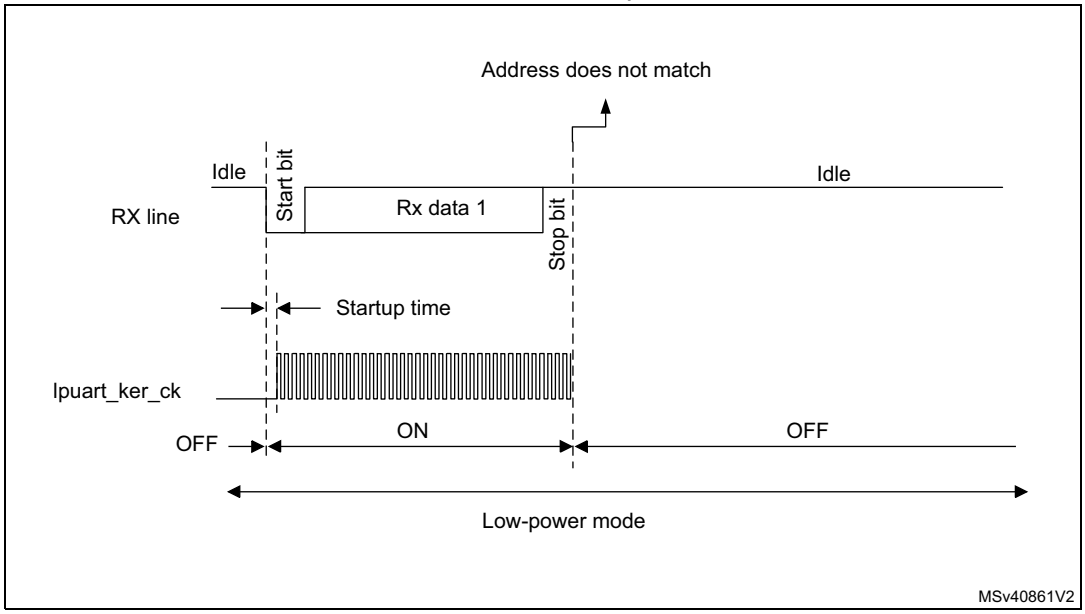


Figure 723 shows the behavior when the wake-up event is not verified.

Figure 723. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 51.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 564: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

51.5 LPUART in low-power modes

Table 566. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 51.3: LPUART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

51.6 LPUART interrupts

Refer to *Table 567* for a detailed description of all LPUART interrupt requests.

Table 567. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 51.3: LPUART implementation* for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

51.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

51.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 50.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 51.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode
When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.
When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: LPUART not able to wake up the MCU from low-power mode.
1: LPUART able to wake up the MCU from low-power mode.
Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled
Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

51.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 50.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 51.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

51.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

51.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

51.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
									rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

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

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

51.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

51.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

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

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

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

51.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.
An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 51.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in active mode
1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE = 1in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by a write to the LPUART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

51.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

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

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 51.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

51.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value.* Refer to *Section 51.3: LPUART implementation.*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

51.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 51.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

51.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 51.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

51.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

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

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

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 51.3: LPUART implementation.

51.7.14 LPUART register map

Table 568. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	LPUART_CR1FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0	0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0				0
0x0C	LPUART_BRR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 568. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

52 Serial peripheral interface (SPI)

52.1 SPI introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

52.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

52.3 SPI implementation

The table below describes the SPI implementation.

Table 569. SPI features

SPI feature	SPI1, SPI2, SPI3 (full-featured instances)	SPI4, SPI5(1), SPI6(1) (full-featured instances)
Data and CRC size	Configurable from 4 to 32 bits	Configurable from 4 to 16 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits	CRC polynomial length configurable from 5 to 17 bits
FIFO size	16x8 bits	8x8 bits
FIFO threshold	1 - 16 data	1 - 16 data
Number of data control (TSIZE)	Up to 65535	Up to 65535
I2S feature	Yes	No
Autonomous in Stop modes with wake-up capability	No	No
Autonomous in LP-Stop and Standby modes with wake-up capability	No	No

1. Available only on STM32H562xI, STM32H563xI and STM32H573xx devices

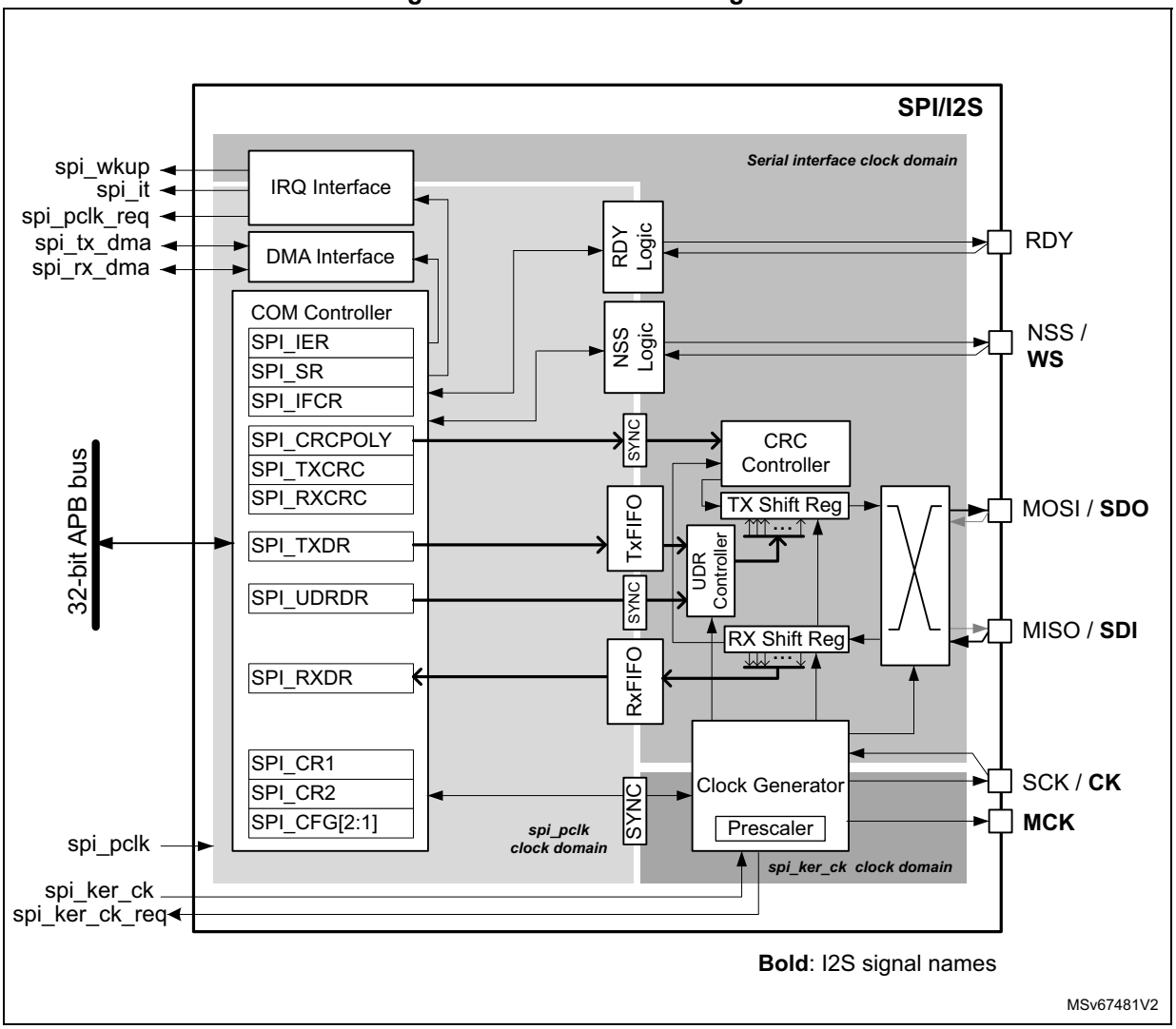
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 573: SPI wake-up and interrupt requests.

52.4 SPI functional description

52.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 724*.

Figure 724. SPI/I2S block diagram



The simplified scheme of *Figure 724* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user

has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 52.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 52.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

52.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 52.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features an additional I2S specific MCK signal. For more details about I2S signals, see *Section 52.9.2*.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 570* and *Table 571* for the list of SPI input / output pins and internal signals.

Table 570. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
MISO/SDI(2)	Input/output	Master data input / slave data output
MOSI/SDO(2)	Input/output	Master data output / slave data input
SCK/CK	Input/output	Master clock output / slave clock input
NSS/WS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output
MCK	Output	I2S master frequency output

1. Refer to the section *Section 52.9.2: Pin sharing with SPI function* for details.
2. Functionality of MOSI/SDO and MISO/SDI pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 571. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request

52.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

52.4.4 Communications between one master and one slave

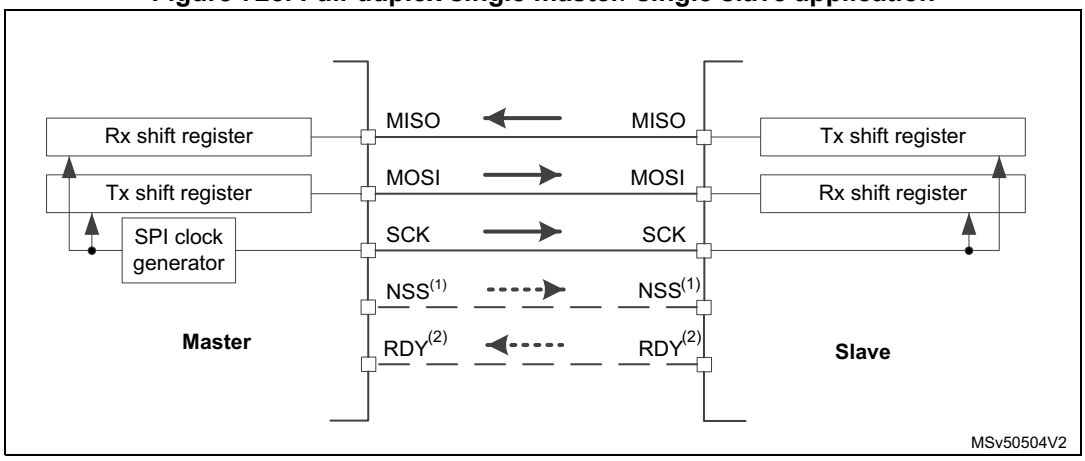
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 725. Full-duplex single master/ single slave application

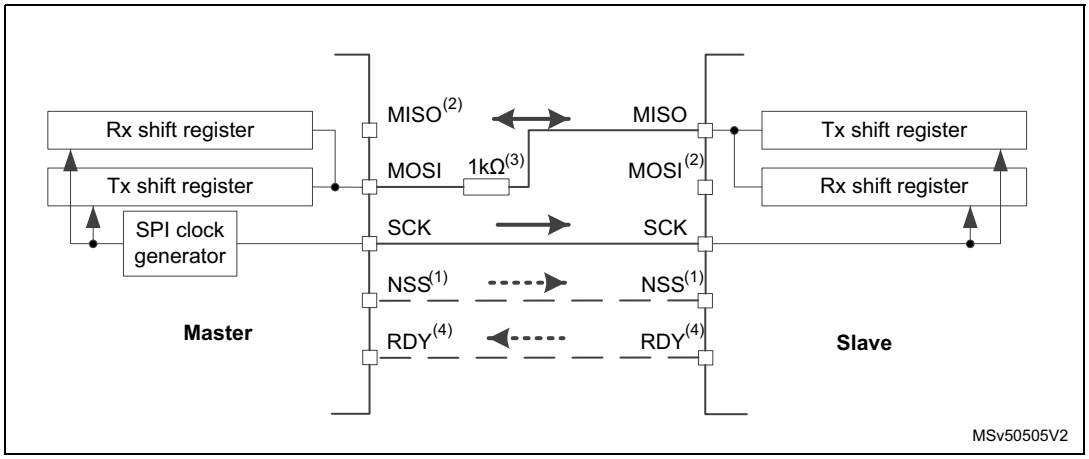


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 52.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 726. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 52.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.
4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 52.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

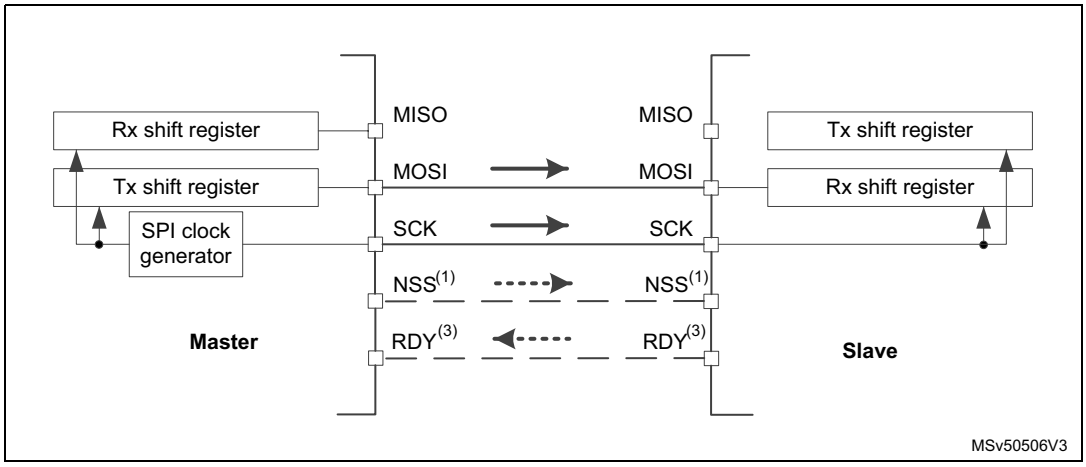
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 52.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the

HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 727. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 52.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

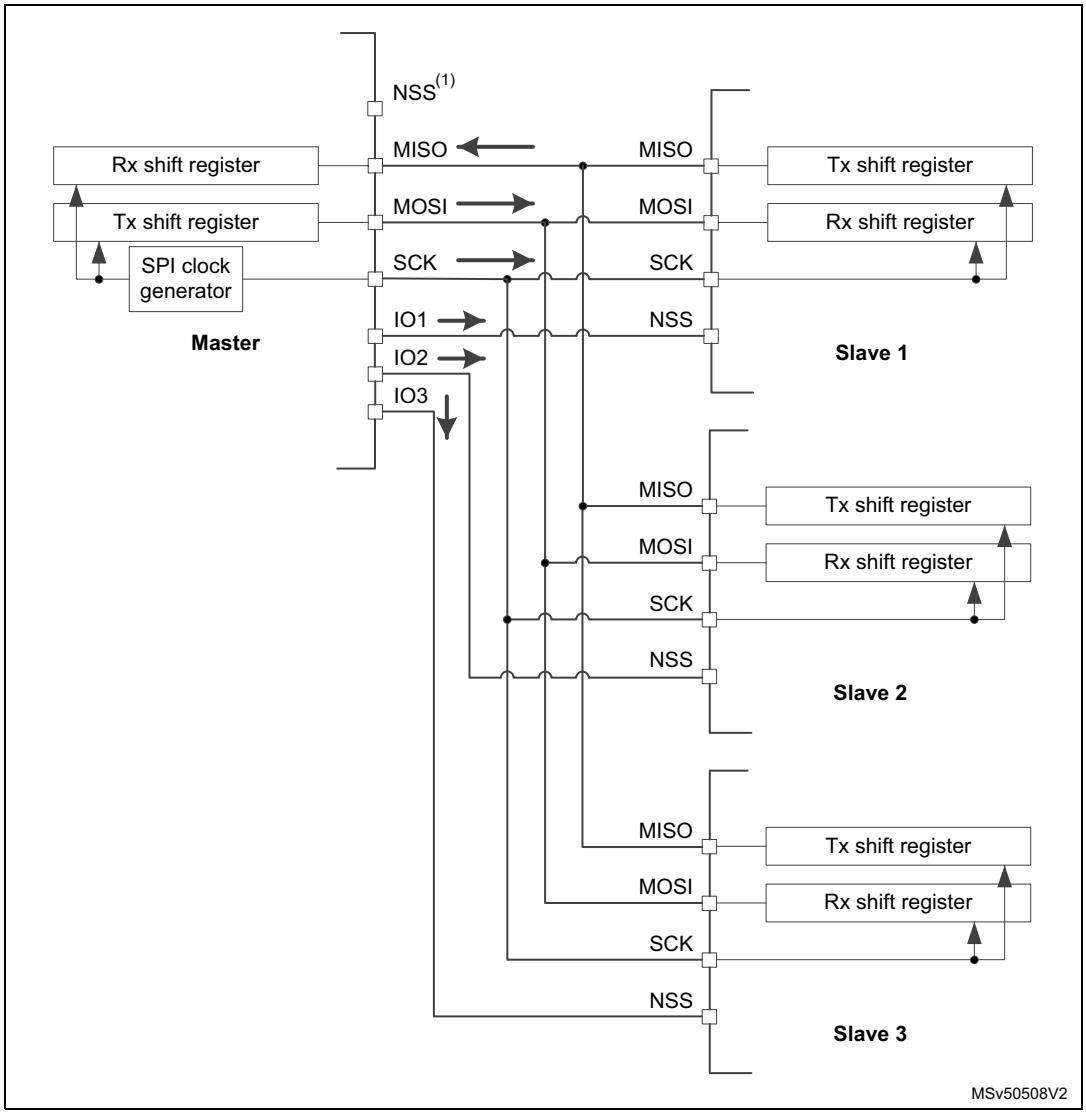
52.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 728.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 728. Master and three independent slaves connected in star topology



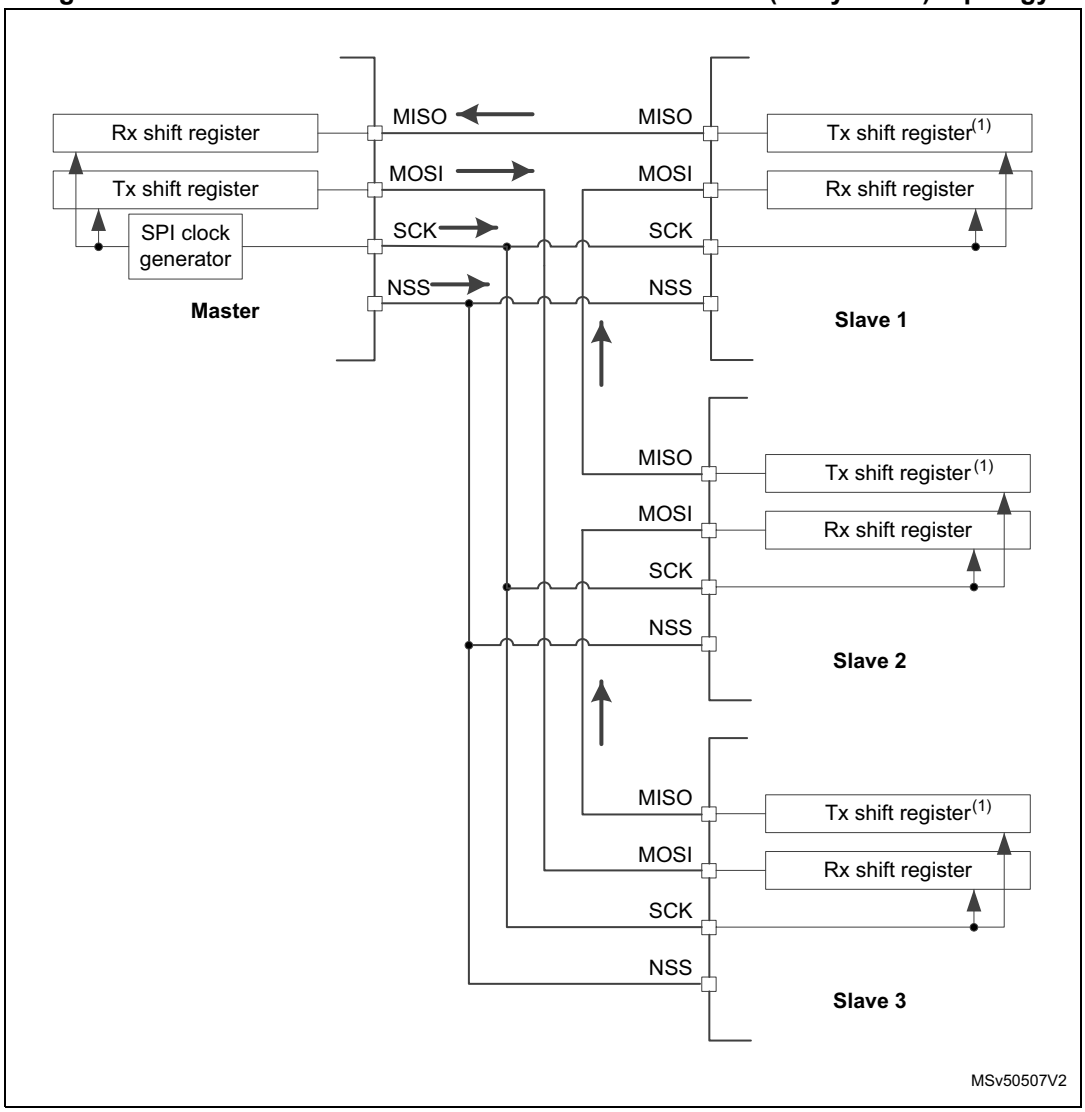
1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 52.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves in time when a circular topology is applied (see *Figure 729*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.

Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 729. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

52.4.6 Multimaster communication

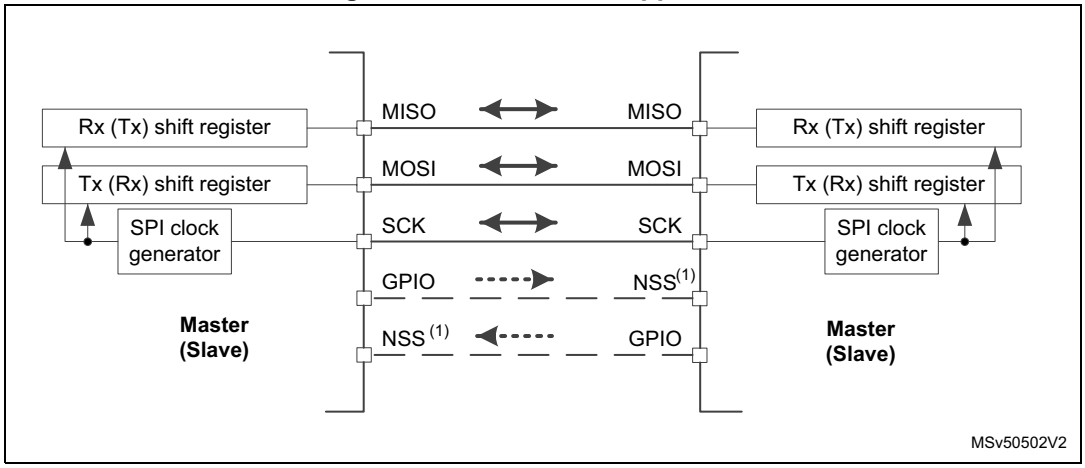
Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is

complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 730. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

52.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

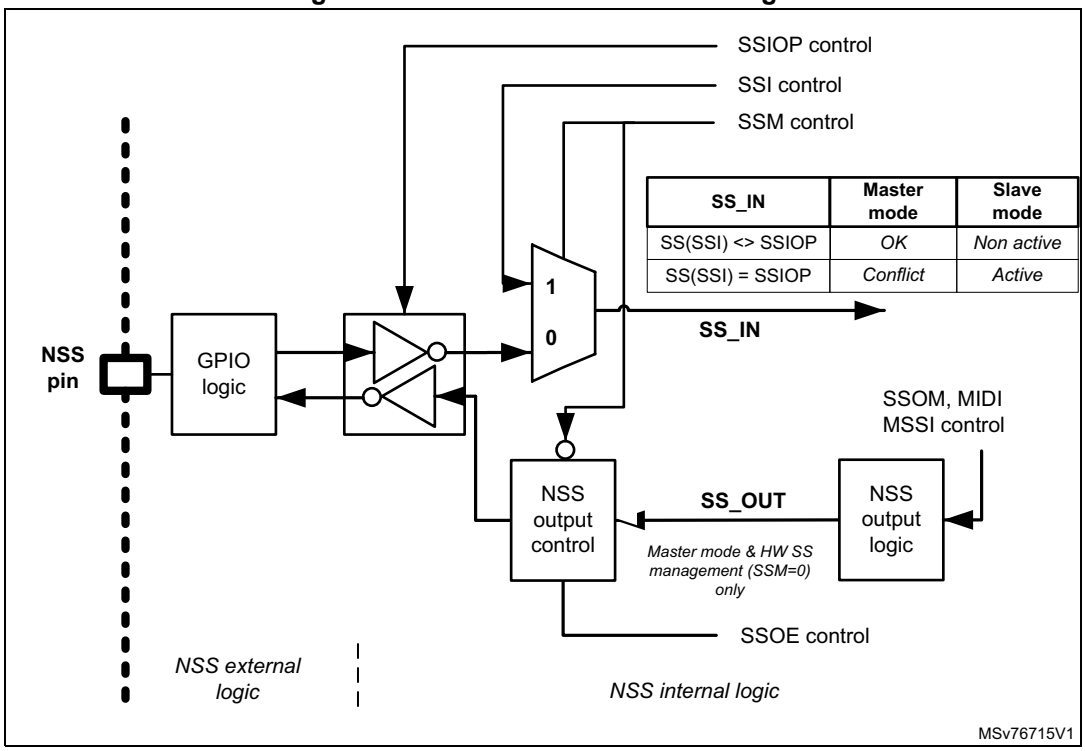
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

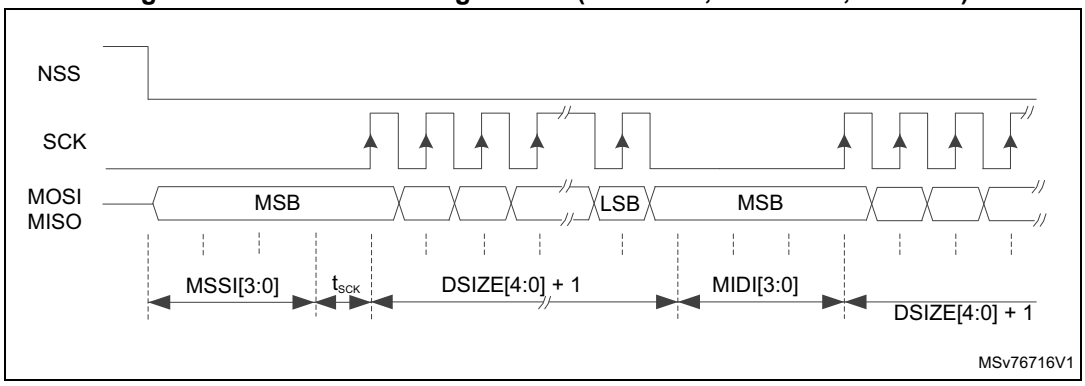
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 731. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 732*).

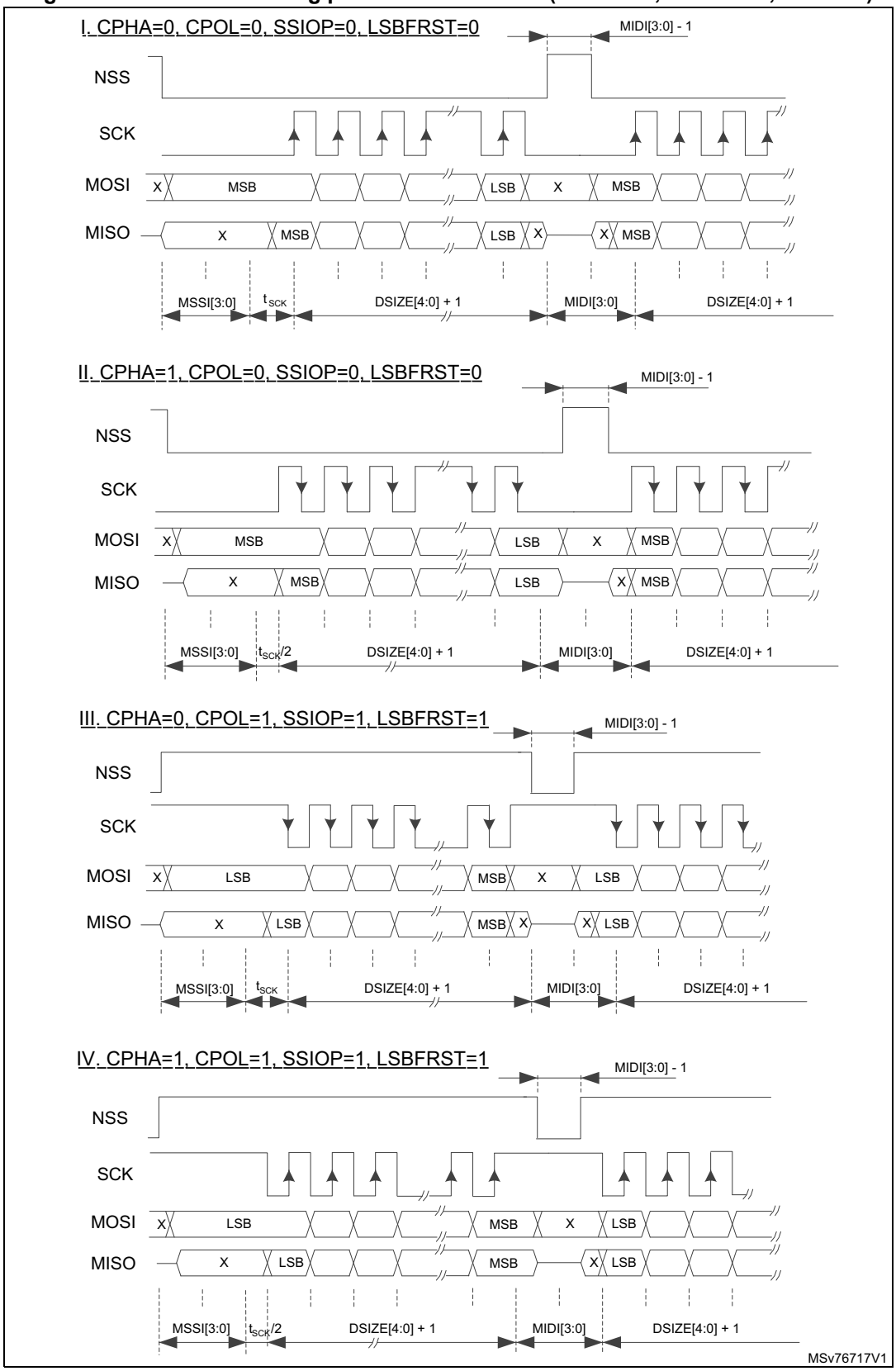
Figure 732. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 733*.

Figure 733. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

52.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

52.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

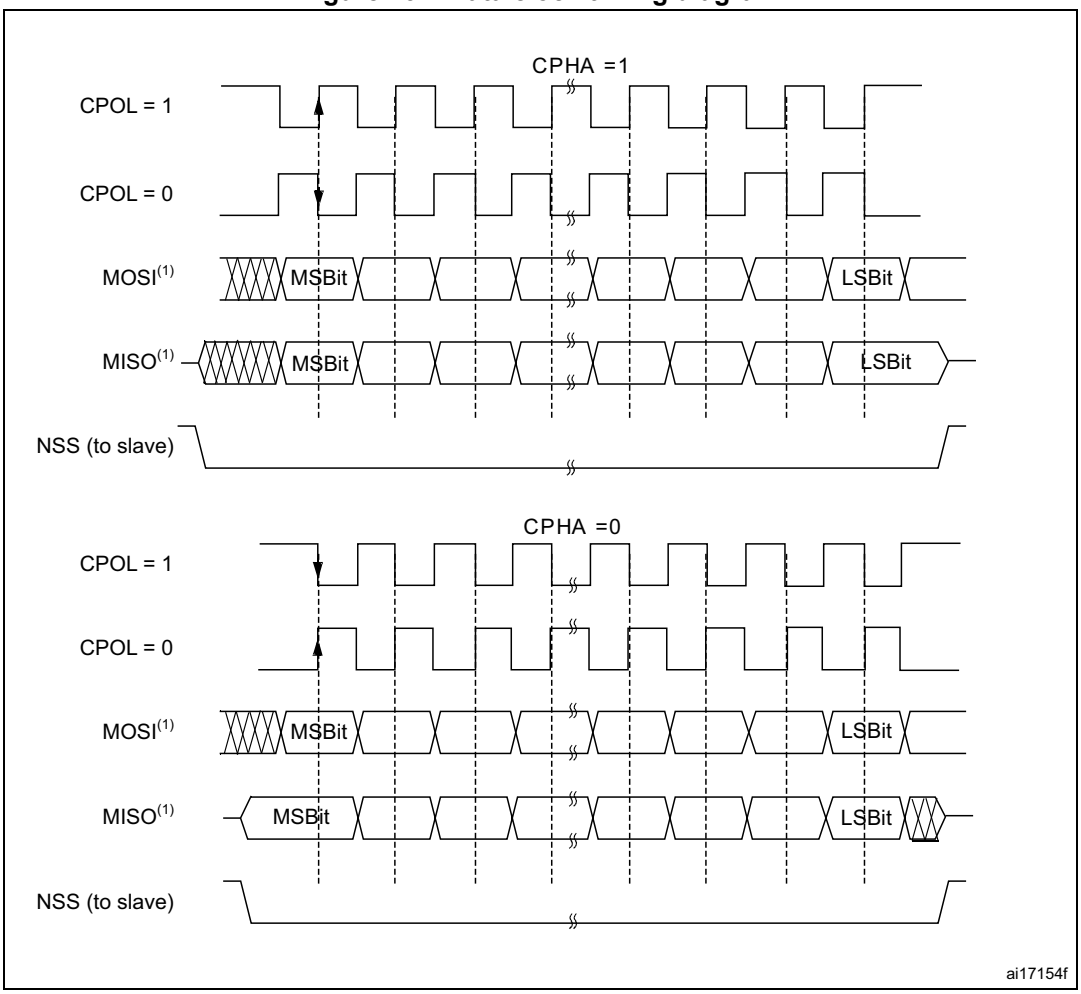
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 734*).

Figure 734 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 734. Data clock timing diagram



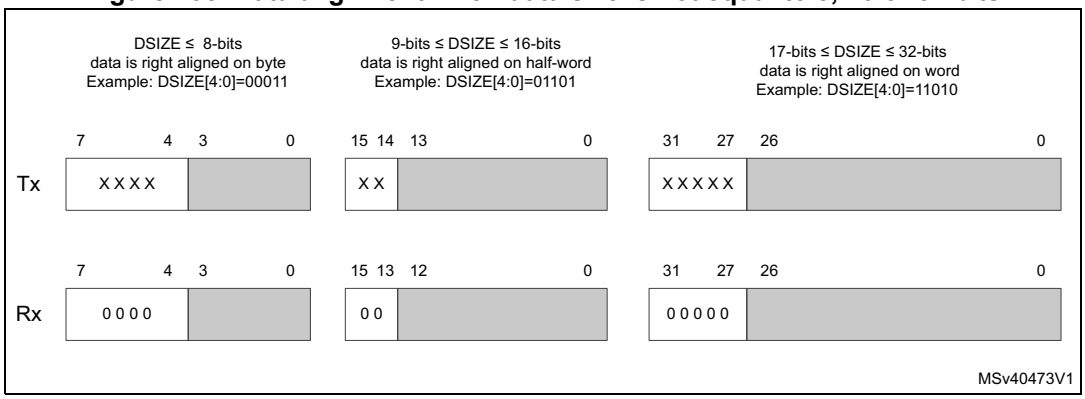
1. The order of data bits depends on the LSBFRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 52.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 735*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 52.4.12: SPI data transmission and reception procedures*.

Figure 735. Data alignment when data size is not equal to 8, 16 or 32 bits



52.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 52.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

52.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 52.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 52.4.14*.

52.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the

content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO

content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

52.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed.

TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.

2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

52.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

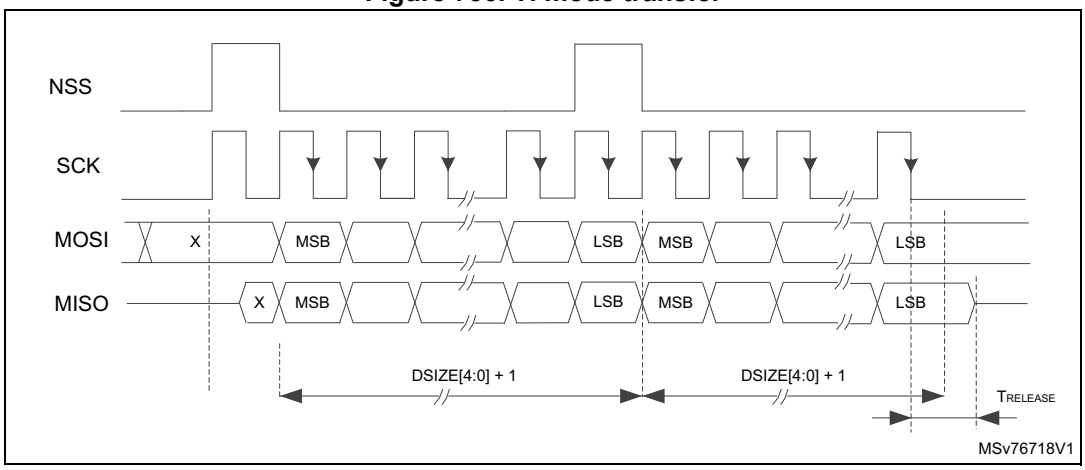
Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

52.5 SPI specific modes and control

52.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 736*.

Figure 736. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

52.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all

data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

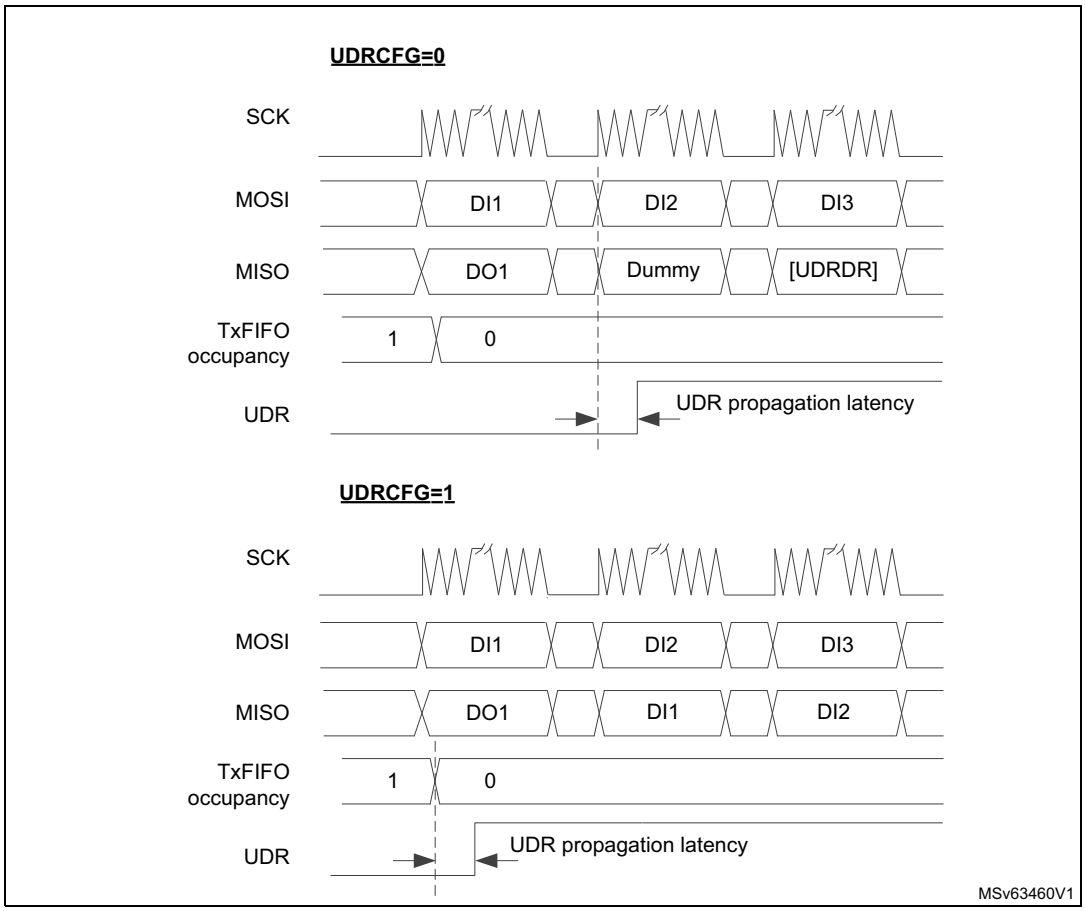
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 737*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 729*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the lastly received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 737. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

52.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 52.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

52.6 SPI in low-power modes

Table 572. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The SPI registers content is kept.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 52.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

52.7 SPI interrupts

Table 573 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 573. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability[1][2]
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC[3]		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 52.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

52.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length can be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, overrun, and frame errors
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

52.9 I2S functional description

52.9.1 I2S general description

The block diagram shown on *Figure 724* also applies for I2S mode.

The SPI/I2S block can work in I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA, and parts of interrupt signaling are shared with the SPI function. The low-power mode function is also available in I2S mode (refer to *Section 52.6: SPI in low-power modes* and *Section 52.10: I2S interrupts*).

a. Not always available, refer to *Section 52.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

52.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: serial data output (mapped on the MOSI pin) to transmit the audio samples in master, and receive the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 2434*.
- SDI: serial data input (mapped on the MISO pin) to receive the audio samples in master, and transmit the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 2434*.
- WS: word select (mapped on the NSS pin). This is the frame synchronization. It is configured as output in master mode, and as input in slave mode.
- CK: serial clock (mapped on the SCK pin). This is the serial bit clock. It is configured as output in master mode, and as input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: master clock (mapped separately) is used when the I2S is configured in master mode.

52.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value to guarantee the behavior of the I2S/PCM function. *Table 574* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 574. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI/I2S control register 2 (SPI_CR2)	-	Set to reset value
SPI/I2S configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI/I2S configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S receive data register (SPI_RXDR)	The complete register	-

Table 574. Bitfields usable in PCM/I2S mode (continued)

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI/I2S transmitter CRC register (SPI_TXCRC)	-	
SPI/I2S receiver CRC register (SPI_RXCRC)	-	
SPI/I2S underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

52.9.4 Slave and master modes

The SPI/I2S block supports master and slave modes for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

To improve the robustness of the SPI/I2S block in slave mode, the peripheral resynchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB-justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected

Note: This resynchronization mechanism is not available for the I2S LSB-justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

52.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits of the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frame formats are supported:

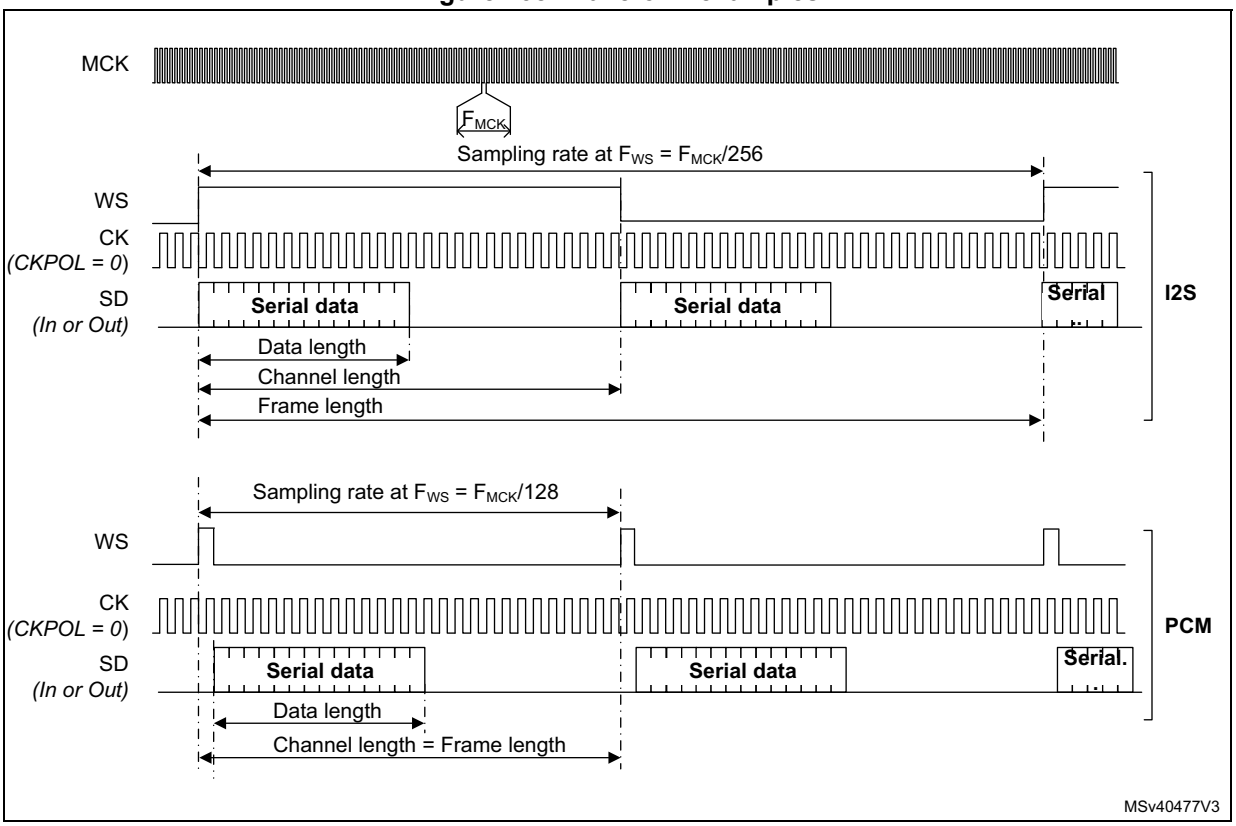
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]
- 32-bit data packed in a 32-bit channel[(a)]

In PCM master mode, three frame formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 738. Waveform examples



For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to *Section : Serial data line swapping* for details about the direction control of serial data lines.

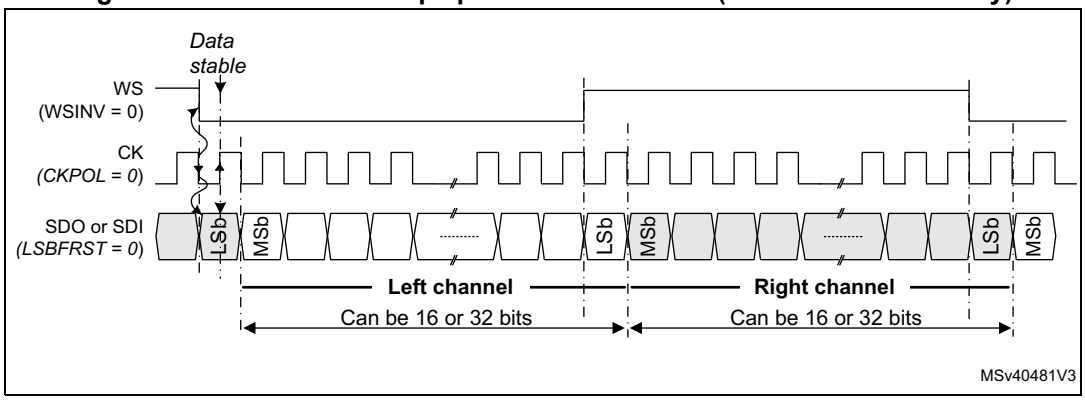
I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

a. Not always available, refer to *Section 52.3: SPI implementation* to check if 24 and 32-bit data widths are supported.

Figure 739. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



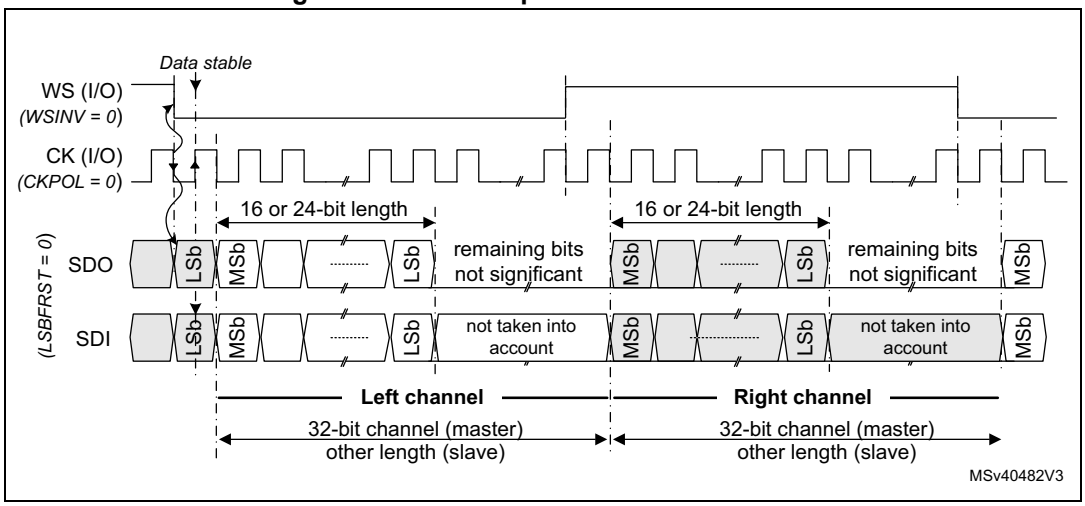
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

CKPOL is cleared to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

Figure 739 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

Figure 740. I2S Philips standard waveforms



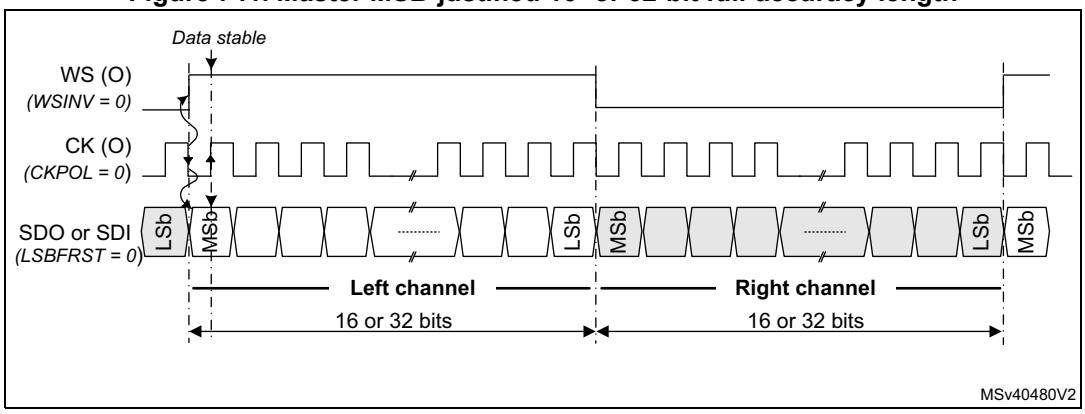
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB-justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high and the right channel if WS is low.

Figure 741. Master MSB-justified 16- or 32-bit full-accuracy length

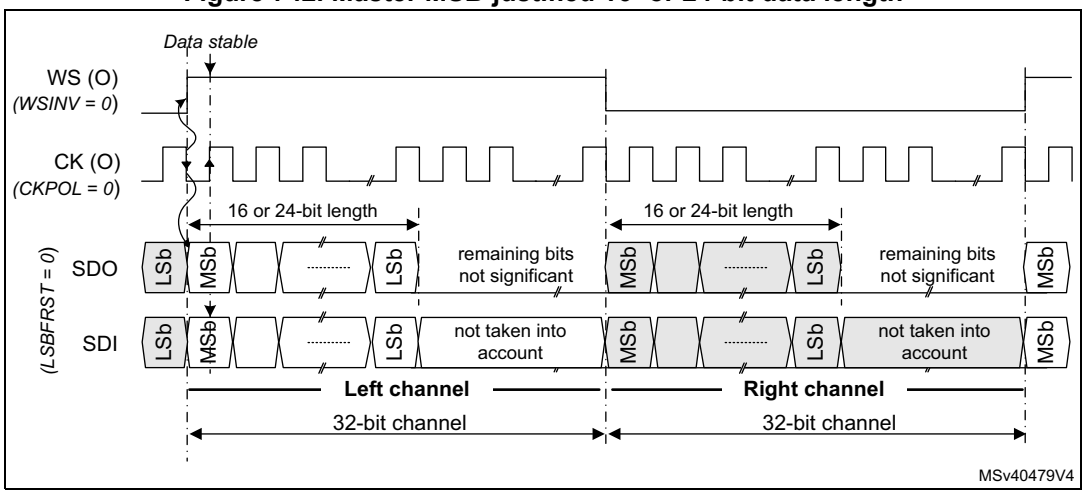


1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* to check the supported data size.

CKPOL is cleared to match the I2S MSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

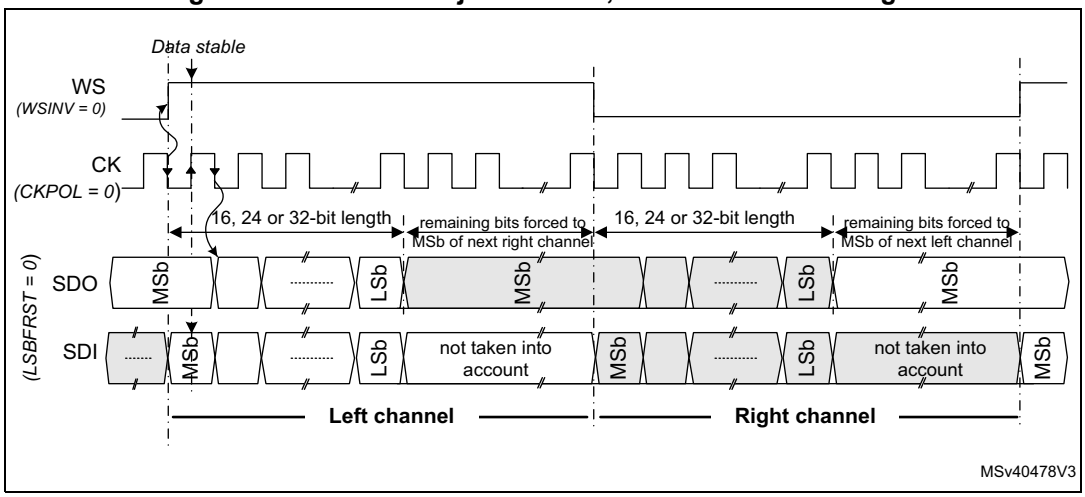
Figure 742. Master MSB-justified 16- or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit mode, the remaining bits are forced to the value of the first bit of the next data to be generated to avoid timing issues (see *Figure 743*).

Figure 743. Slave MSB-justified 16-, 24- or 32-bit data length



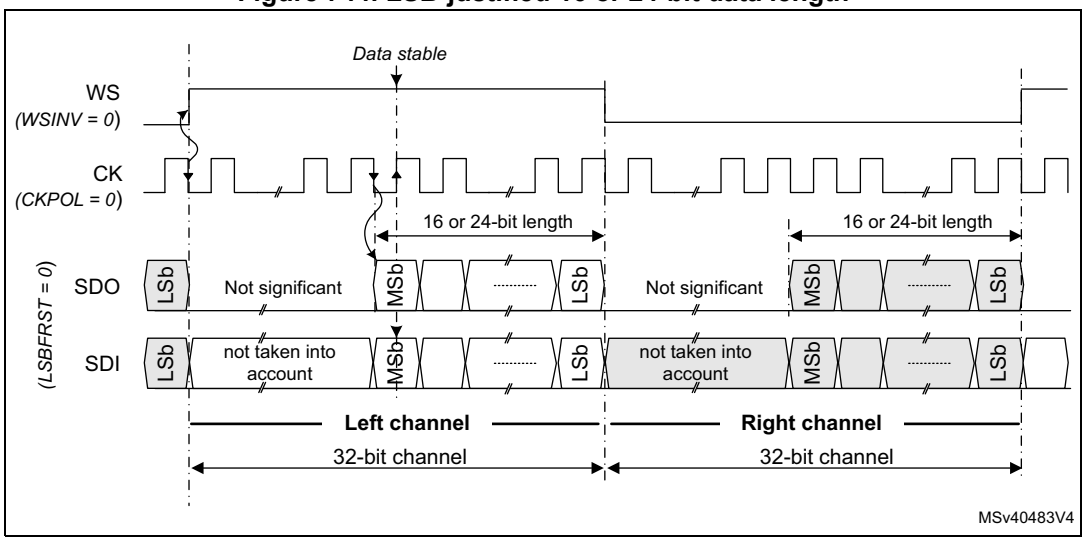
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* to check the supported data size.

LSB-justified standard

This standard is similar to the MSB-justified standard in master mode (no difference for the 16- and 32-bit full-accuracy frame formats). The LSB-justified 16- or 32-bit full-accuracy format gives similar waveforms to MSB-justified mode (see *Figure 741*) because the channel and data have the same length.

Note: In the LSB-justified format, only 16- and 32-bit channel lengths are supported in master and slave mode. This is because it is not possible to transfer properly the data if the channel length is not known by the transmitter and receiver side.

Figure 744. LSB-justified 16 or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* to check the supported data size.

CKPOL is cleared in order to match the I2S LSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit of SPI_I2SCFGR register.

In PCM long frame:

- The assertion time of the WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

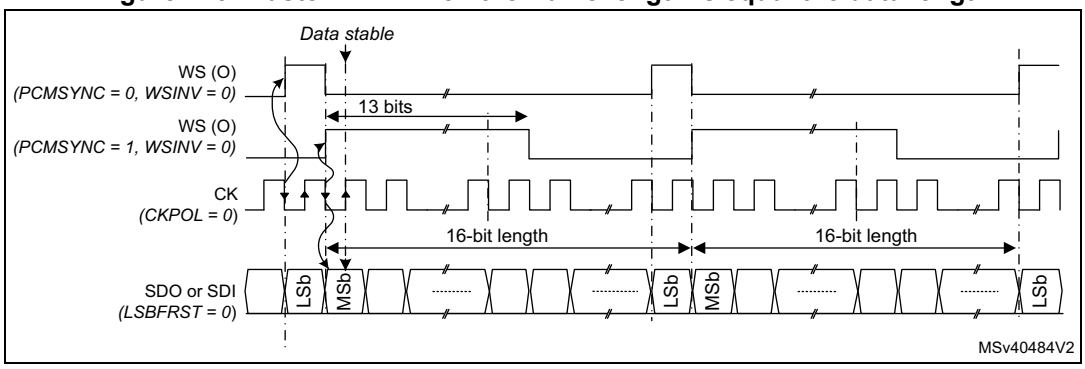
In PCM short frame:

- The assertion time of the WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:

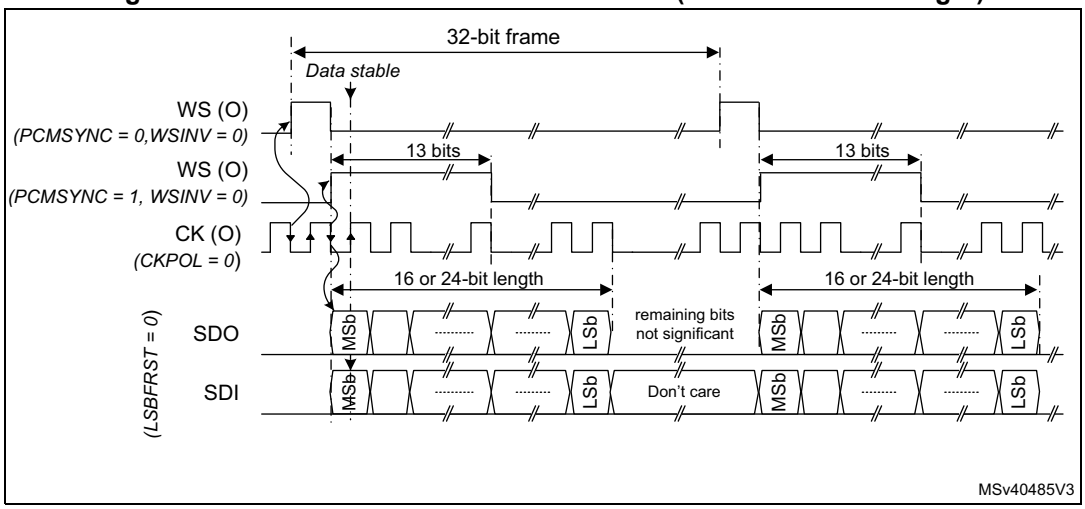
- The first data bit is MSb or LSb depending on the LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to the WSINV bit. See *Control of the WS inversion* for information concerning the handling of WS signal.

Figure 745. Master PCM when the frame length is equal the data length



A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

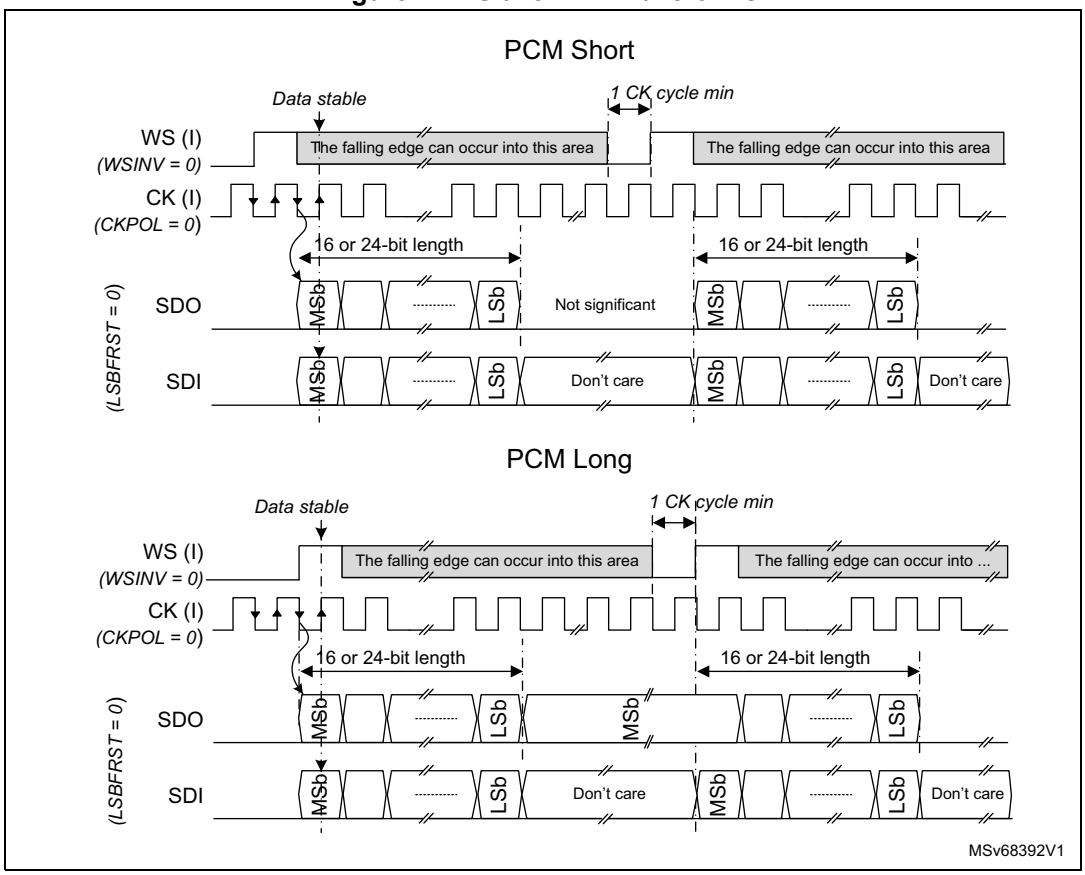
Figure 746. Master PCM standard waveforms (16 or 24-bit data length)



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* to check the supported data size.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 747*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 747. Slave PCM waveforms



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* to check the supported data size.

Note: In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits is forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

52.9.6 Additional serial interface flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for the I2S LSB-justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI/I2S configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB-justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB-justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located in the *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI/I2S configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 575 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS also depends on the protocol selected.

Table 575. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR must not be set when the SPI is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of *SPI/I2S configuration register 2 (SPI_CFG2)*, and on the slave/master mode. *Table 576* gives details on this feature.

Table 576. Serial data line swapping

Direction	IOSWP	Master mode		Slave mode	
		Input line	Output line	Input line	Output line
RX	0	SDI	-	SDO	-
	1	SDO	-	SDI	-
TX	0	-	SDO	-	SDI
	1	-	SDI	-	SDO
Full-duplex	0	SDI	SDO	SDO	SDI
	1	SDO	SDI	SDI	SDO

52.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write from/to the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

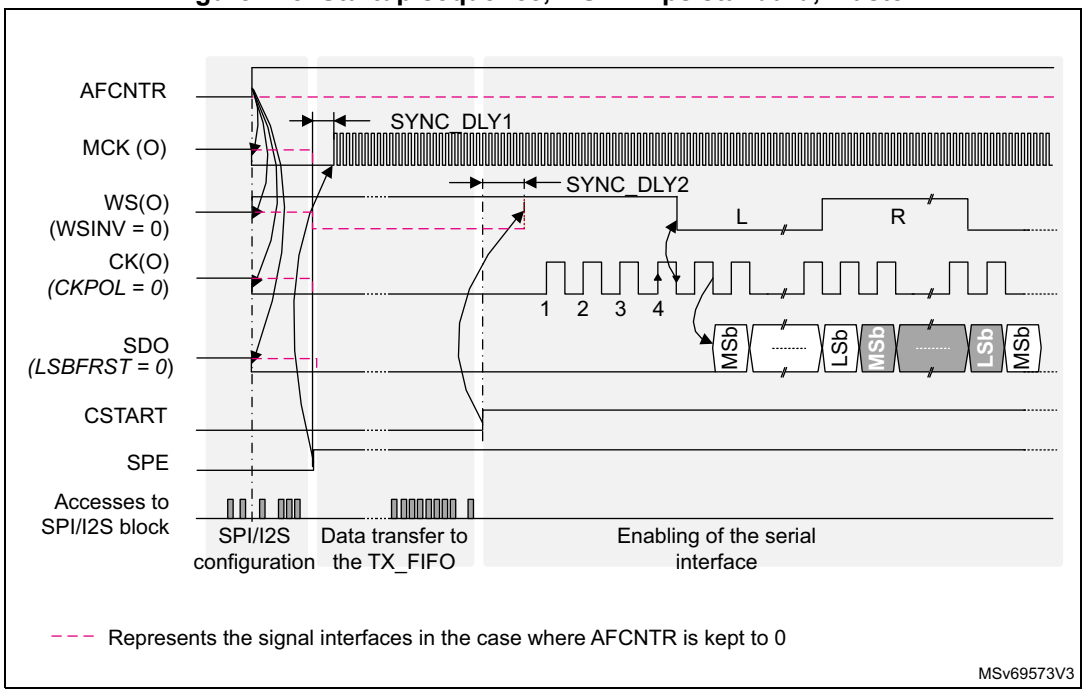
When the application wants to use the SPI/I2S block the user has to proceed as follows:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 52.9.16* for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC, and UDRC bits of the SPI_IFCR register. Note that if the flag SUSP is not cleared (via the SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO. The generation of MCK can also be started when SPE goes to 1.
5. Set bit CSTART to activate the serial interface.

As shown in *Figure 748*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1

and the TxFIFO is not empty. Note that the CK bit clock is activated four rising edges before the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 748. Startup sequence, I2S Philips standard, master



1. As shown in the figure, MCK can be enabled as soon as the bit SPE is set. It is generated after a synchronization delay (SYNC_DLY1). See MCK generation in *Section 52.9.9: Clock generator* for more information.
2. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

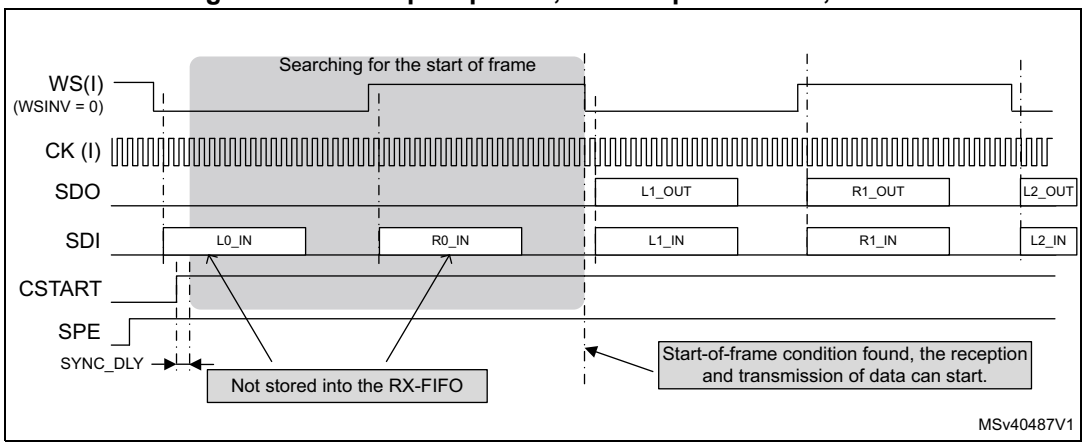
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about three periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For the I2S Philips standard, the start-of-frame condition is a falling edge of the WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of the WS signal. The transmission/reception starts at the rising edge of WS for MSB-aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 749 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 749. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after two periods of CK clock (SYNC_DLY).

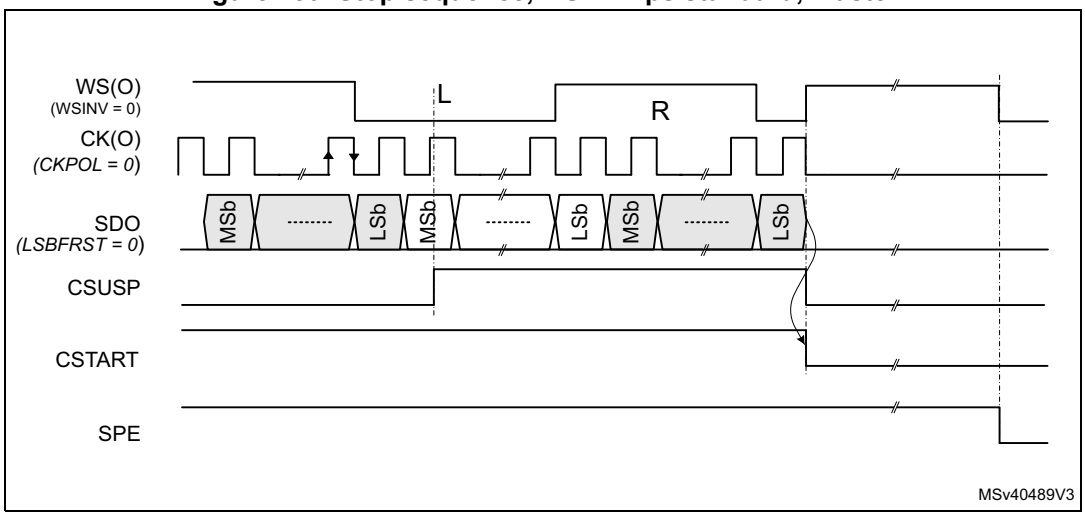
52.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

Figure 750 shows an example of stop sequence in the case of master mode. The CSUSP bit is set during the transmission of the left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear the SPE bit.

Figure 750. Stop sequence, I2S Philips standard, master

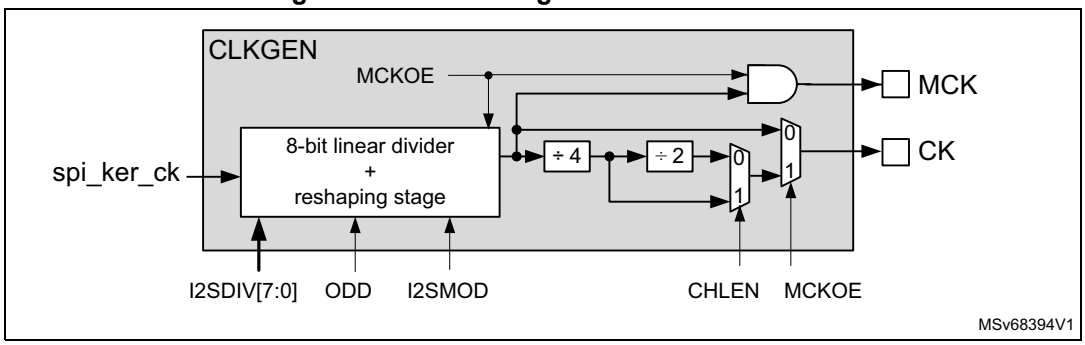


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

52.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the frame synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 751. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN, and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there are no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.

I2SDIV can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value I2SDIV = 1 is not allowed.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: When {(2 x I2SDIV) + ODD} is odd, the duty cycle of the MCK or the CK signal is not 50%. Care must be taken when an odd ratio is used: it can impact margin on setup and hold time. For example if {(2 x I2SDIV) + ODD} = 5, then the duty cycle can be 40%.

Table 577 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK is generated when the following conditions are met:

- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,
- SPE must be set

Table 577. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192

Table 577. CLKGEN programming examples for usual I2S frequencies (continued)

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

52.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

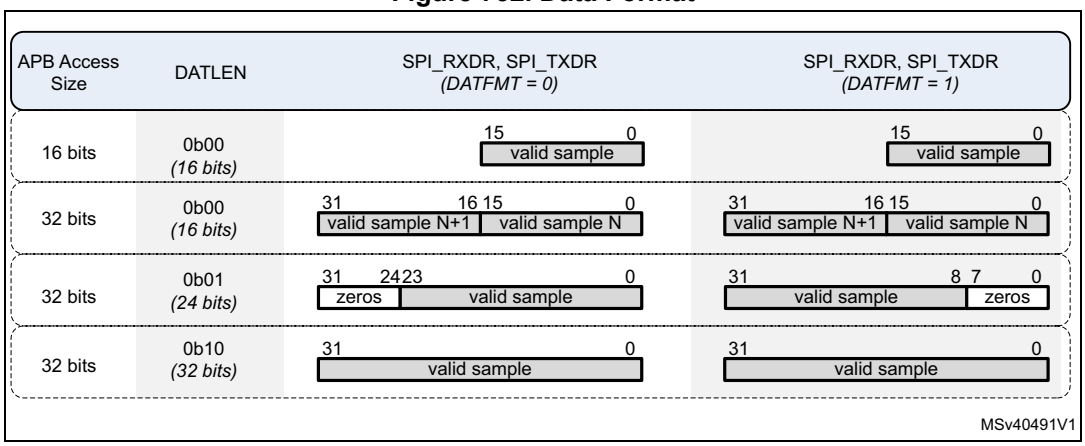
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 752 shows the allowed settings between APB access sizes, DATFMT, and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 752. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 52.3: SPI implementation* to check the supported data size.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to *Section 52.9.11* and *Section 52.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes three basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[a],
- 4 samples, if the DATLEN = 2 (32 bits)[a].

52.9.11 FIFO status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus, FTHLV new data to be transmitted can be written to SPI_TXDR. The TXP flag is reset when the number of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 52.10* for additional information on the interrupt function in I2S mode.

52.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

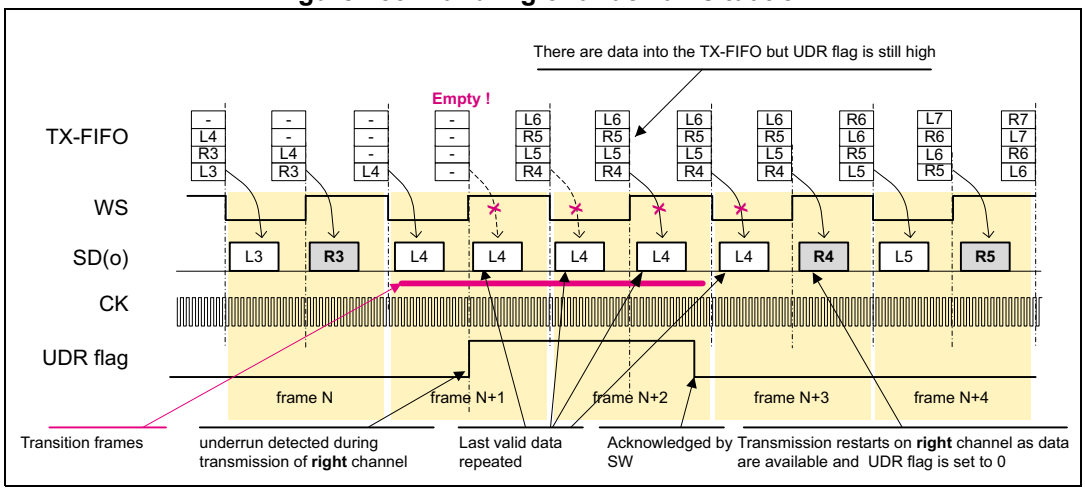
a. Not always available, refer to *Section 52.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral replays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software, and
- When the restart condition is met:
  - If the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - If the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

Figure 753. Handling of underrun situation



When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO, and the UDR flag is cleared.

The UDR flag can trigger an interrupt if the UDRIE bit of the SPI_IER register is set. The UDR bit is cleared by setting the UDRC bit of the SPI_IFCR register.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK, and MCK signals are not gated.

Due to resynchronization, any change of the UDR flag is taken into account by the hardware after at least two periods of CK clock.

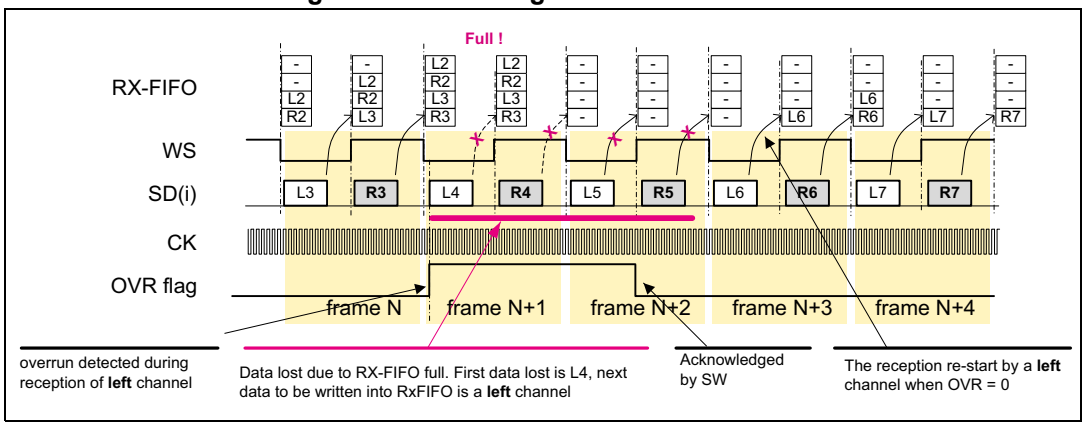
52.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.

In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO, and the OVR flag is cleared. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 754. Handling of overrun situation



When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, and the OVR flag is cleared, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by setting the OVRC bit of the SPI_IFCR register.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK, and MCK signals are not gated.

52.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

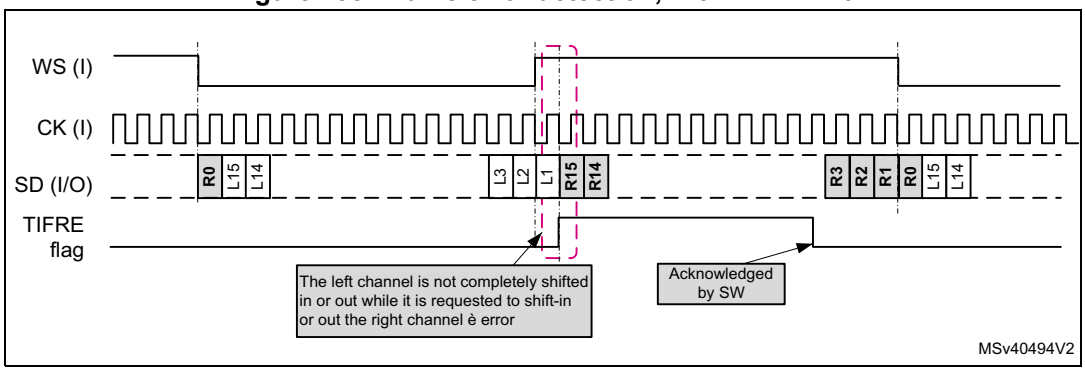
- A frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurs at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.

Figure 755 shows an example of frame error detection. The SPI/I2S block is in slave mode and the number of bit clock periods for the left channel are not enough to shift-in or shift-out the data. The figure shows that the ongoing transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 755. Frame error detection, with FIXCH = 0

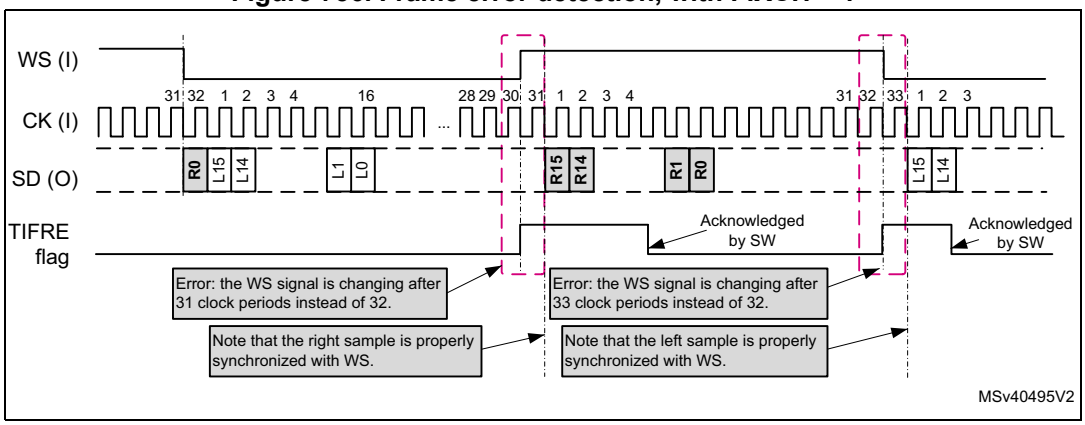


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 756*, in this mode, the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 755* and *Figure 756* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 756. Frame error detection, with FIXCH = 1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

52.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of the SPI_CFG1 register.

In receive mode, the DMA interface is working as follows:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follows:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

52.9.16 Programing examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set I2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example, if a threshold of two audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. Clear all interrupt enable fields located in the SPI_IER register.
8. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of the SPI_IFCR register.
9. Set the SPE bit, then the MCK is generated. In the example presented here, the TxFIFO is not filled-up as soon as SPE = 1. The application can then configure the external audio codec via an I2C interface even if audio samples to transmit are not yet available. An *Alternative sequence* is proposed hereafter.
10. When the application wants to start the data transfer:
    - If the data transfer uses DMA:
    a) Program the DMA peripheral: two channels, one for RX and one for TX.
    b) Initialize the memory buffer with valid audio samples for TX path.
    c) Enable the DMA channels.
    d) Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
    e) Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled up.
    - If the data transfer is done via interrupt:

      Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Alternative sequence

- Steps 1 to 8 are similar to the previous sequence.
- If the data transfer uses DMA:
  - Program the DMA peripheral: two channels, one for RX and one for TX
  - Initialize the memory buffer with valid audio samples for the TX path
  - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
  - Enable the DMA channels,

- If the data transfer is done via interrupt:
  - Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
- Set SPE, as soon as this bit is set to one the following actions may happen:
  - If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
  - If the DMA transfer is enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
  - The MCK is generated. The application can then configure the external audio codec via an I2C interface.
- Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Stop procedure in master mode

1. Set the bit CSUSP, in order to stop ongoing transfers
2. Check the value of the CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So, we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example if a threshold of two audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   - Program the DMA peripheral: one RX channel
   - Enable the DMA channel,
   - In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.

8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally, the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop procedure in slave mode

1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

52.10 I2S interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wake-up event signal (**spi_wkup**) can be generated according to the events described in the *Table 578*.

Interrupt events can be enabled and disabled separately.

Table 578. I2S interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/Interrupt clearing method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
SPI	TxFIFO threshold reached	TXP	When the TxFIFO contains less than FTHLV empty locations	Yes	Yes	No
	RxFIFO threshold reached	RXP	When the RxFIFO contains less than FTHLV samples			
	Overrun error	OVR	Write OVRC to 1			
	Underrun error	UDR	Write UDRC to 1			
	Frame error flag	TIFRE	Write TIFREC to 1		No	

52.11 SPI/I2S registers

52.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (33-bit or 17-bit) CRC polynomial configuration

0: Full size (33-bit or 17-bit) CRC polynomial is not used

1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode

In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 52.9.8: Stop sequence*.

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero.

SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

52.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

52.11.3 SPI/I2S configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw						rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator

0: bypass is disabled

1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting

000: SPI master clock/2

001: SPI master clock/4

010: SPI master clock/8

011: SPI master clock/16

100: SPI master clock/32

101: SPI master clock/64

110: SPI master clock/128

111: SPI master clock/256

Note: MBR setting is considered at slave working at TI mode, too (see Section 52.5.1: TI mode).

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable

0: CRC calculation disabled

1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared

Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.

The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.

The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

Bit 15 **TXDMAEN:** Tx DMA stream enable

0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable

0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

For more details see *Figure 737: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in the SPI_UDRDR register

1: Slave repeats lastly received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8bit, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

52.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Note: This bit can be also used in PCM and I2S modes.

The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped
Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness

Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.

0000: no delay

0001: 1 clock cycle period delay

...

1111: 15 clock cycle periods delay

Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

52.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 9 **MODFIE**: mode Fault interrupt enable
0: MODF interrupt disabled
1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable
0: TIFRE interrupt disabled
1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable
0: CRC interrupt disabled
1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable
0: OVR interrupt disabled
1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

52.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus .
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is re-enabled or by writing 1 to the CRCEC bit of the SPI_IFCR optionally.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is re-enabled or by writing 1 to the OVRC bit of the SPI_IFCR optionally.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is re-enabled or by writing 1 to the UDRC bit of the SPI_IFCR optionally.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only. In I2S/PCM mode, (when available) this flag applies to master and slave mode.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

52.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear

Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

52.11.8 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

52.11.9 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

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

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

52.11.10 SPI/I2S polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

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

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

52.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

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

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

52.11.12 SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

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

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

52.11.13 SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

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

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

52.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x050

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit, which needs to be kept at 0.

This register is reserved for instances not supporting I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

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

Bit 25 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

Refer to *Section 52.9.9: Clock generator* for details

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
I2SDIV can take any values except the value 1, when ODD is also equal to 1.
Refer to *Section 52.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: The data inside the SPI_RXDR or SPI_TXDR register are right aligned
1: The data inside the SPI_RXDR or SPI_TXDR register are left aligned.

Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB- or LSB-justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB- or LSB-justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 52.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I²S standard selection
For more details on I²S standards, refer to *Section 52.9.5: Supported audio protocols*
00: I²S Philips standard.
01: MSB-justified standard (left justified)
10: LSB-justified standard (right justified)
11: PCM standard

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - Full Duplex
101: master - Full Duplex
others: not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

52.11.15 SPI/I2S register map

Table 579. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0						0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	Reserved	Res.																															
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 579. SPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SPI_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	Reset value							0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 52.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

53 Serial audio interface (SAI)

Available only on STM32H562/563/573xx devices.

53.1 SAI introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many audio protocols can be addressed thanks to SAI “free protocol mode”. The free protocol mode allows the application to define the audio frame shape, number of slots, slot size, data size, and so on. For example, the SAI supports I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, and AC’97 protocols. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has it own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIs to work synchronously.

53.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Possible synchronization between multiple SAIs
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode

  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception
- Interrupt sources when enabled:
  - Errors
  - FIFO requests
- 2-channel DMA interface

53.3 SAI implementation

Table 580. STM32H5 SAI features (1)

SAI features	SAI1(2)	SAI2(2)
I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97	X	X
FIFO size	8 words	8 words
SPDIF	X	X
PDM	X(3)	-

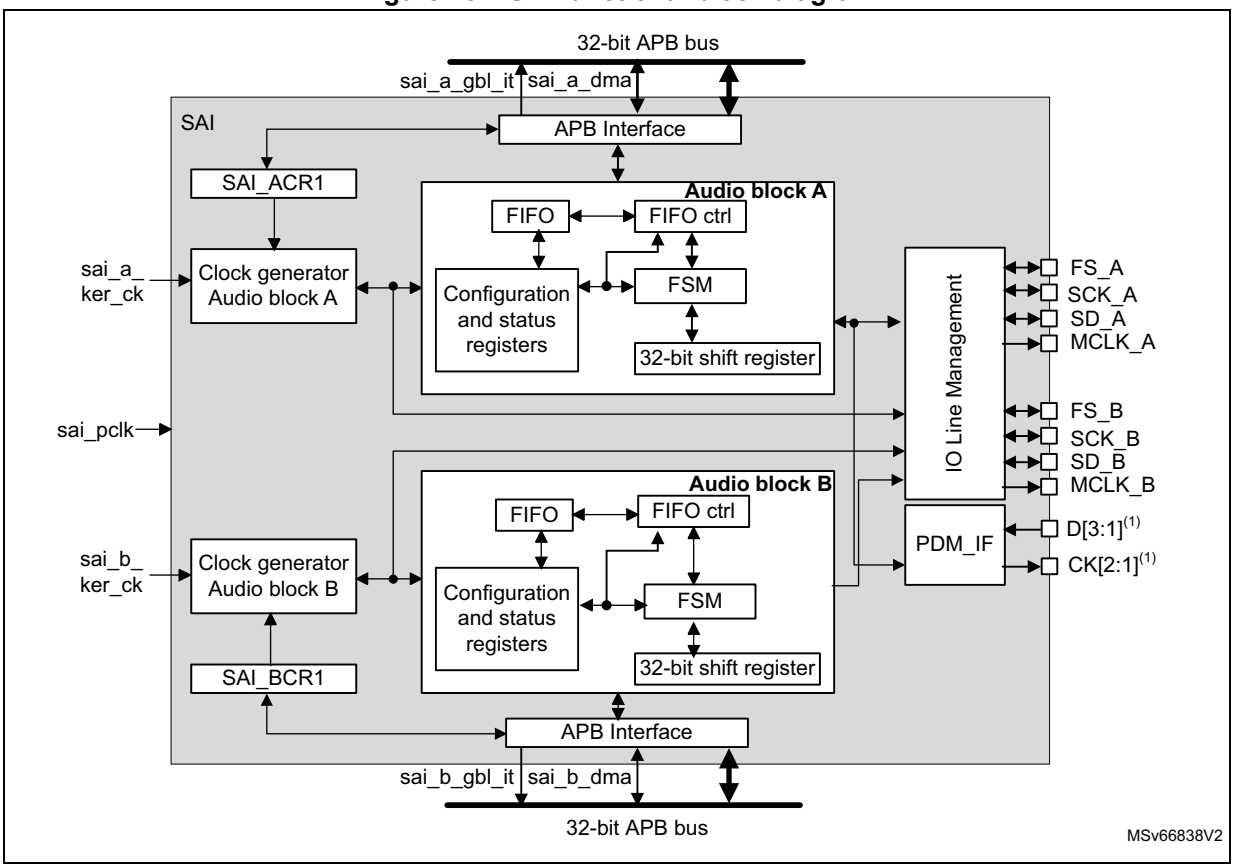
1. ‘X’ = supported, ‘-’ = not supported.
2. Available only on STM32H562xx, STM32H563xx and STM32H573xx devices.
3. Only signals D[3:1], and CK[2:1] are available.

53.4 SAI functional description

53.4.1 SAI block diagram

Figure 757 shows the SAI block diagram while *Table 581* and *Table 582* list SAI internal and external signals.

Figure 757. SAI functional block diagram



1. These signals might not be available for all SAI instances. Refer to *Section 53.3: SAI implementation* for details.

The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC'97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B subblock.

53.4.2 SAI pins and internal signals

Table 581. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests
sai_sync_out_sck, sai_sync_out_fs	Output	Internal clock and frame synchronization output signals exchanged with other SAI blocks
sai_sync_in_sck, sai_sync_in_fs	Input	Internal clock and frame synchronization input signals exchanged with other SAI blocks
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock
sai_pclk	Input	APB clock

Table 582. SAI input/output pins

Pin name	Pin type	Comments
SAI_SCK_A/B	Input/output	Audio block A/B bit clock
SAI_MCLK_A/B	Output	Audio block A/B master clock
SAI_SD_A/B	Input/output	Data line for block A/B
SAI_FS_A/B	Input/output	Frame synchronization line for audio block A/B
SAI_CK[2:1]	Output	PDM bitstream clock(1)
SAI_D[3:1]	Input	PDM bitstream data(1)

1. These signals might not be available in all SAI instances. Refer to *Section 53.3: SAI implementation* for details.

53.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with another SAI interface or with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio block can use a different audio protocol. When PRTCFG[1:0] of the SAI_xCR1 register is set to 0, the free protocol mode is selected and each SAI subblock can emulate I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, or AC’97 protocols.

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold *(*FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

53.4.4 SAI synchronization mode

There are two levels of synchronization, either at audio subblock level or at SAI level.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

External synchronization

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follows:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblocks is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.
2. The SAI which receives the synchronization signals, has to select which SAI provides the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

Note: The SYNCIN[1:0] and SYNCOUT[1:0] bits are located in the SAI_GCR register, and the SYNCEN bits in the SAI_xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as ‘Reserved’ must not be used.

Table 583. External synchronization selection

Block instance	SYNCIN = 1	SYNCIN = 0
SAI1	SAI2 sync.	Reserved
SAI2	Reserved	SAI1 sync.

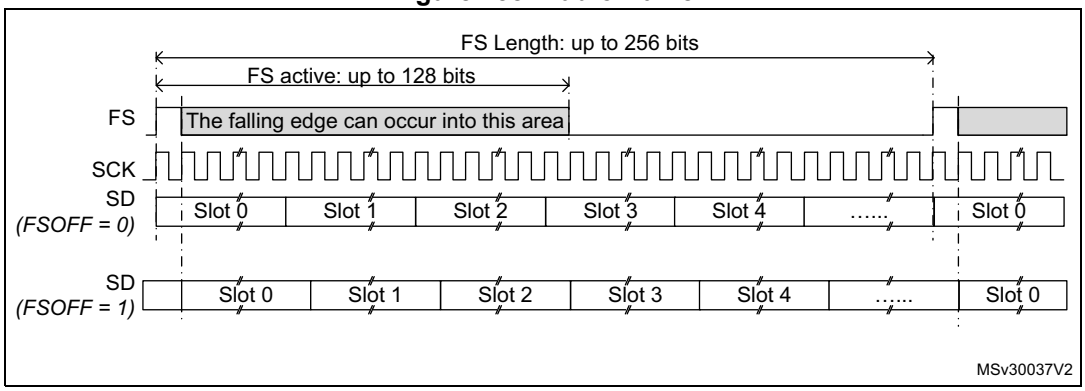
53.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

53.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. *Figure 758* illustrates this flexibility.

Figure 758. Audio frame



In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC'97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z

depending on the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 53.4.8: SAI clock generator*"

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the frame synchronization signal during an ongoing audio frame. In this case, an error is generated. For more details, refer to *Section 53.4.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 53.4.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in *Section 53.4.14: Error flags*), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the frame synchronization signal. The length can be set from 1- to 128-bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

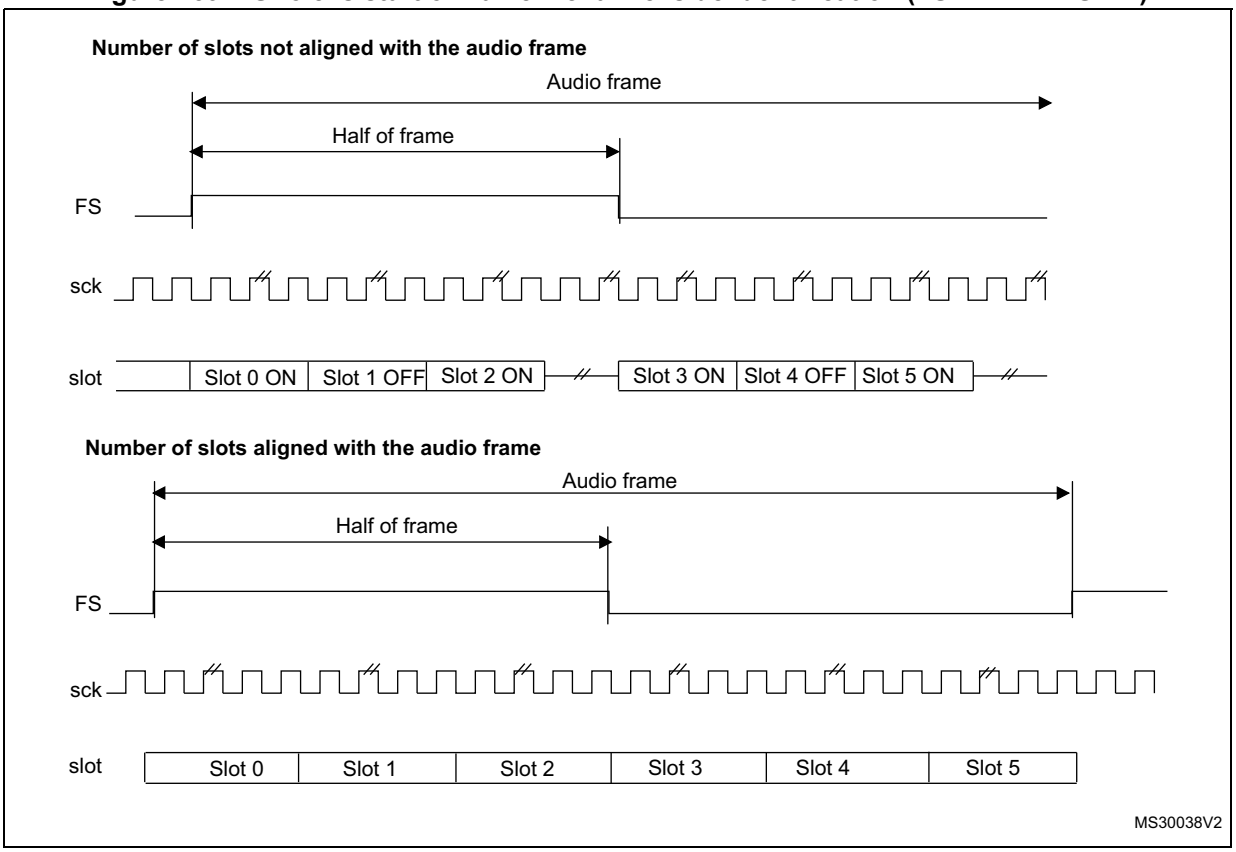
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like, for instance, the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 759. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

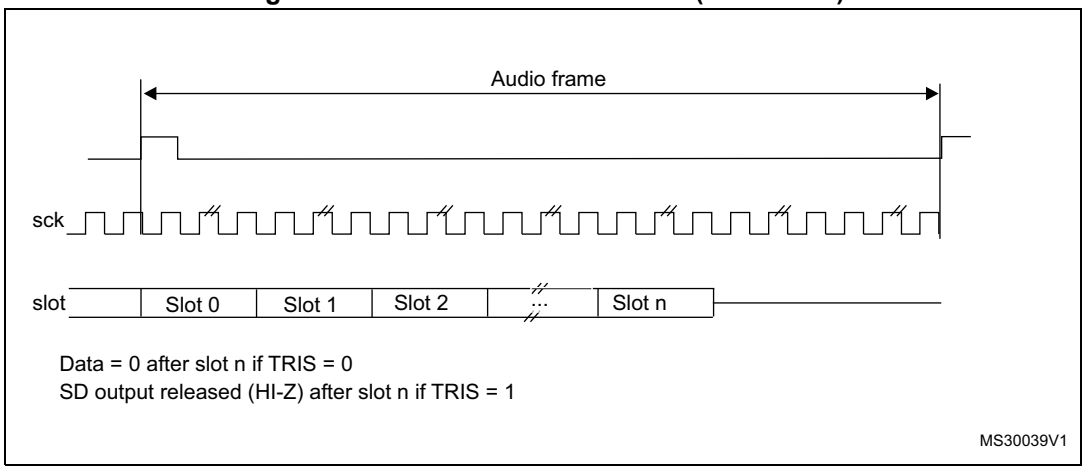


1. The frame length must be even.

If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 760. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

53.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

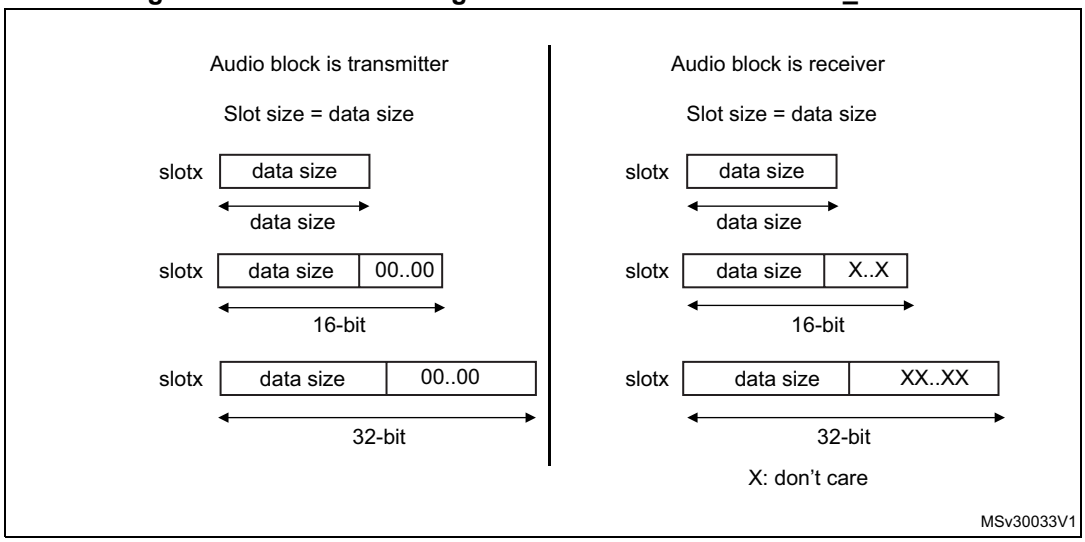
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

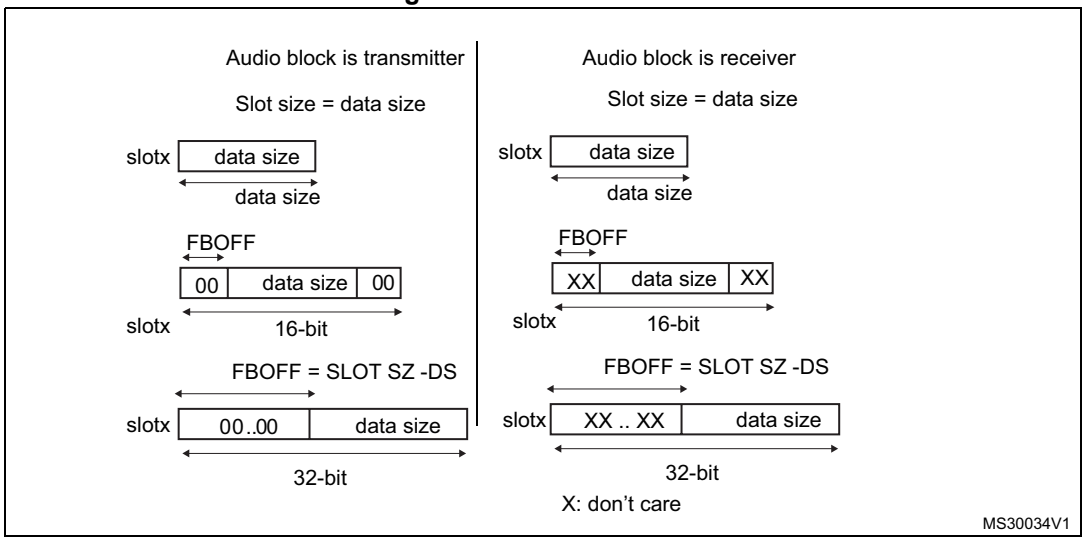
The slot size is also configurable as shown in *Figure 761.* The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 761. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 762. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤(SLOTSZ - DS)
- DS ≤SLOTSZ
- NBSLOT x SLOTSZ ≤FRL (frame length)

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 53.4.11: AC'97 link controller*.

53.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

Table 584 shows MCLK_x activation conditions.

Table 584. MCLK_x activation conditions

MCLKEN	NODIV	SAIEN for block x	MCLK_x
0	X	0	Disabled
1			Enabled
0	1	1	Disabled
1			Enabled
X	0		Enabled

Note: MCLK_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

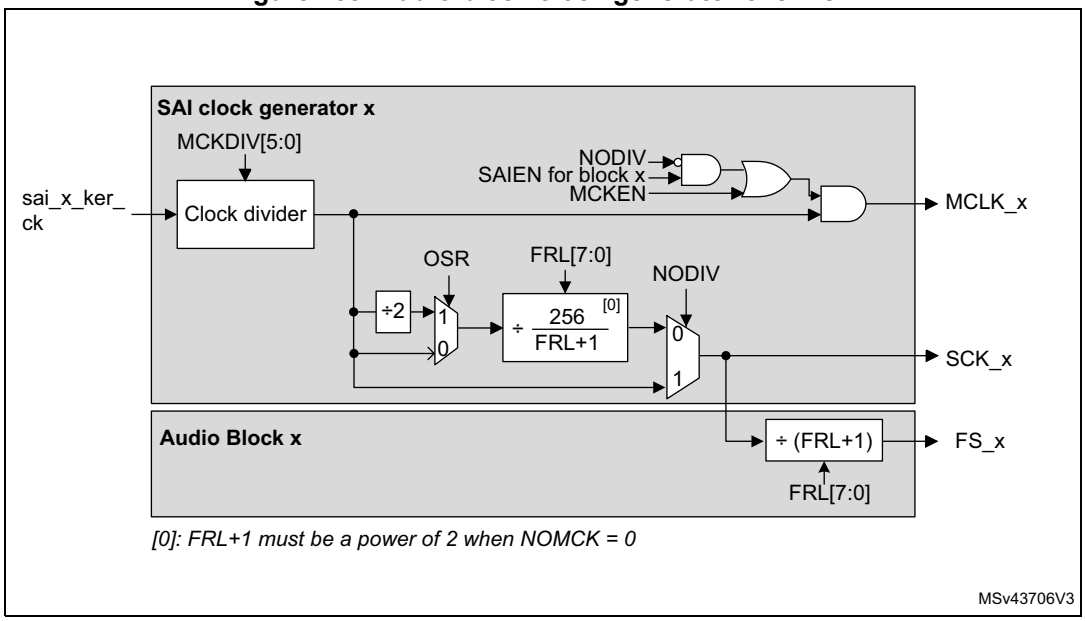
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 763 illustrates the architecture of the audio block clock generator.

Figure 763. Audio block clock generator overview



The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.

Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

- $F_{MCLK_x} = 256 \times F_{FS_x}$ if OSR = 0
- $F_{MCLK_x} = 512 \times F_{FS_x}$ if OSR = 1

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

$$F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^8.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following formula:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 585 gives programming examples for 48, 96, and 192 kHz.

Table 585. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/ F_{FS}	FRL (1)	OSR	NODIV	MCKEN	MCKDIV[5:0]	Audio sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	1	0 or 1	192 kHz
		512	2^N-1	1	0	1	2	96 kHz
		512	2^N-1	1	0	1	4	48 kHz
		256	2^N-1	0	0	1	2	192 kHz
		256	2^N-1	0	0	1	4	96 kHz
		256	2^N-1	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

53.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This

interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

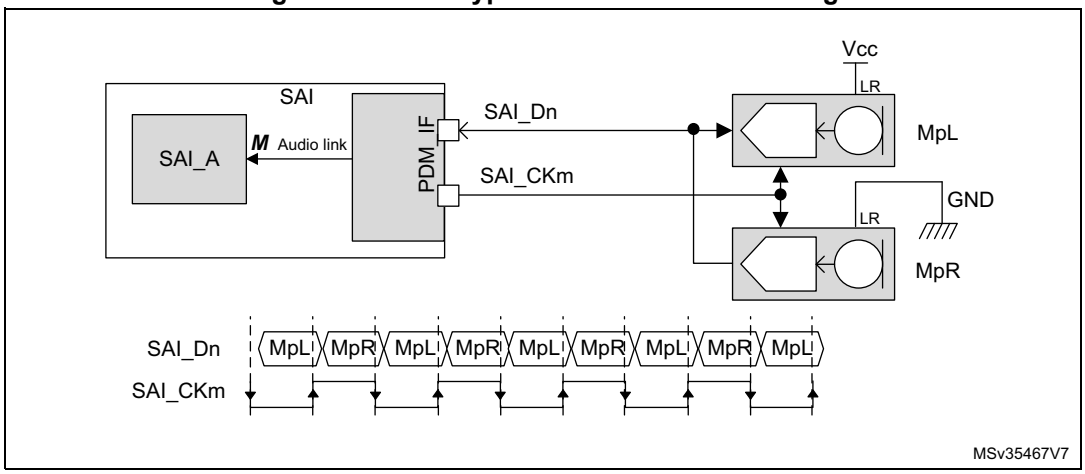
The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

53.4.10 PDM interface

The PDM (pulse density modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to *Section 53.3: SAI implementation*).

Figure 764 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 764. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the serial audio interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

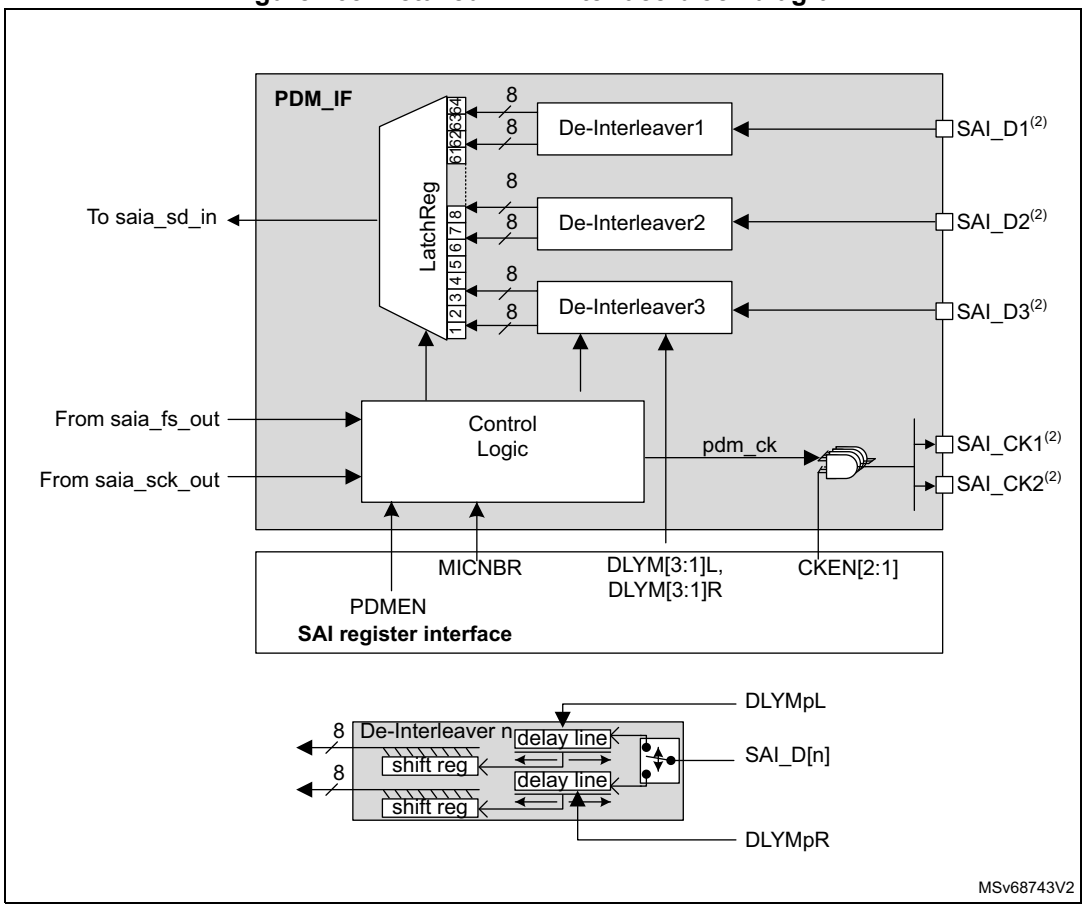
The processing performed into the PDM interface is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the serial audio interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A through the data line of the serial audio interface.

Figure 765 below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

Figure 765. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to *Section 53.3: SAI implementation* for details.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.

To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

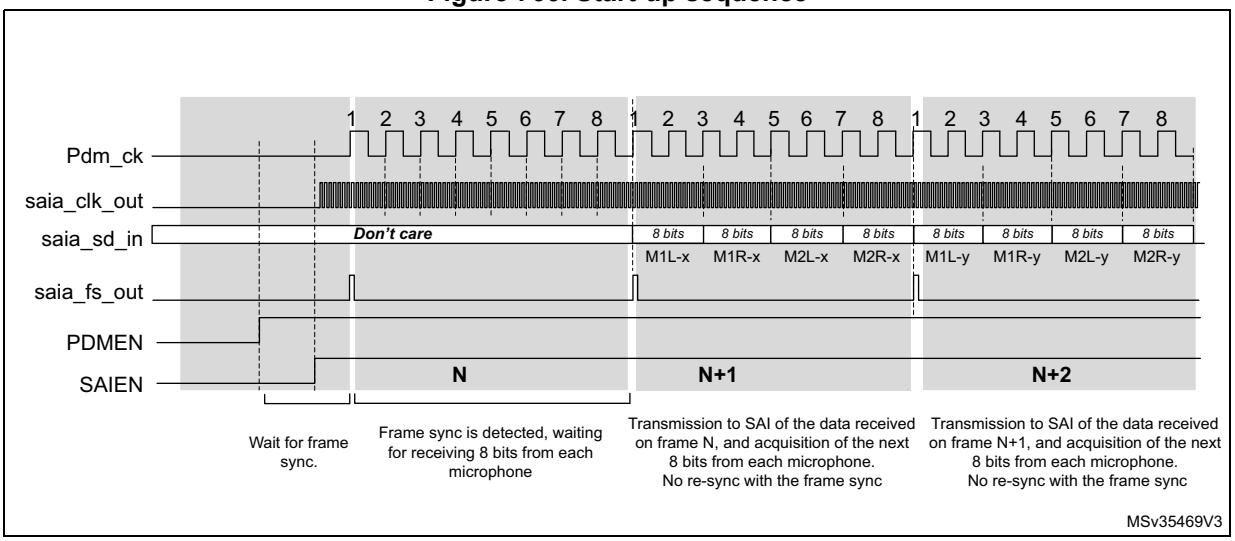
1. Configure SAI_A in TDM master mode (see *Table 586*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first two frames received on SAI_ADR are invalid and must be dropped.

Startup sequence

Figure 766 shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the serial audio interface.

Figure 766. Start-up sequence



SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

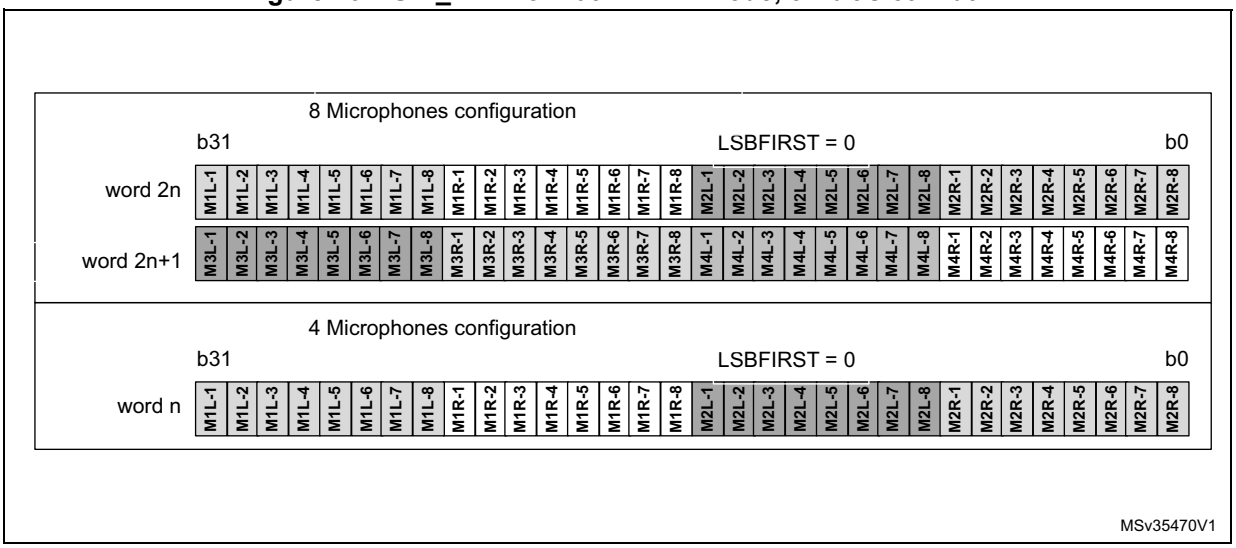
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 767*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 767. SAI_ADR format in TDM mode, 32-bit slot width

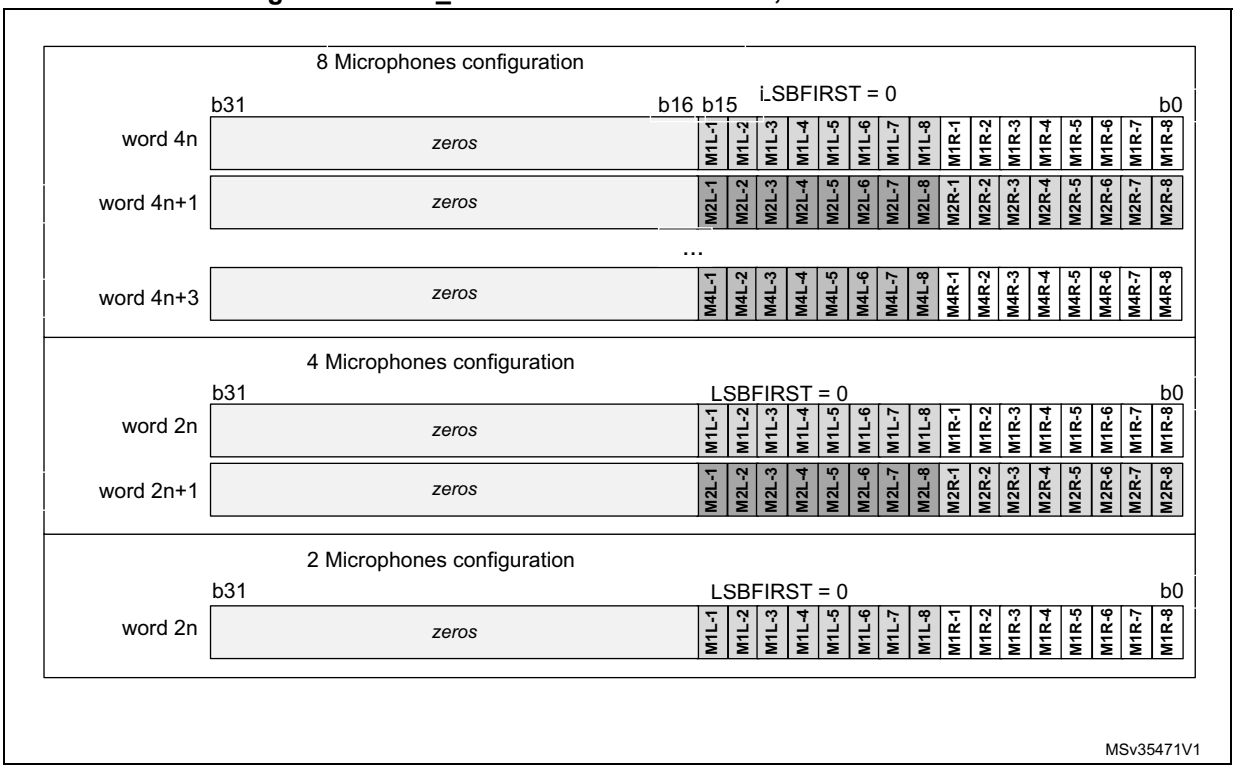


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 768*.

  For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 768. SAI_ADR format in TDM mode, 16-bit slot width

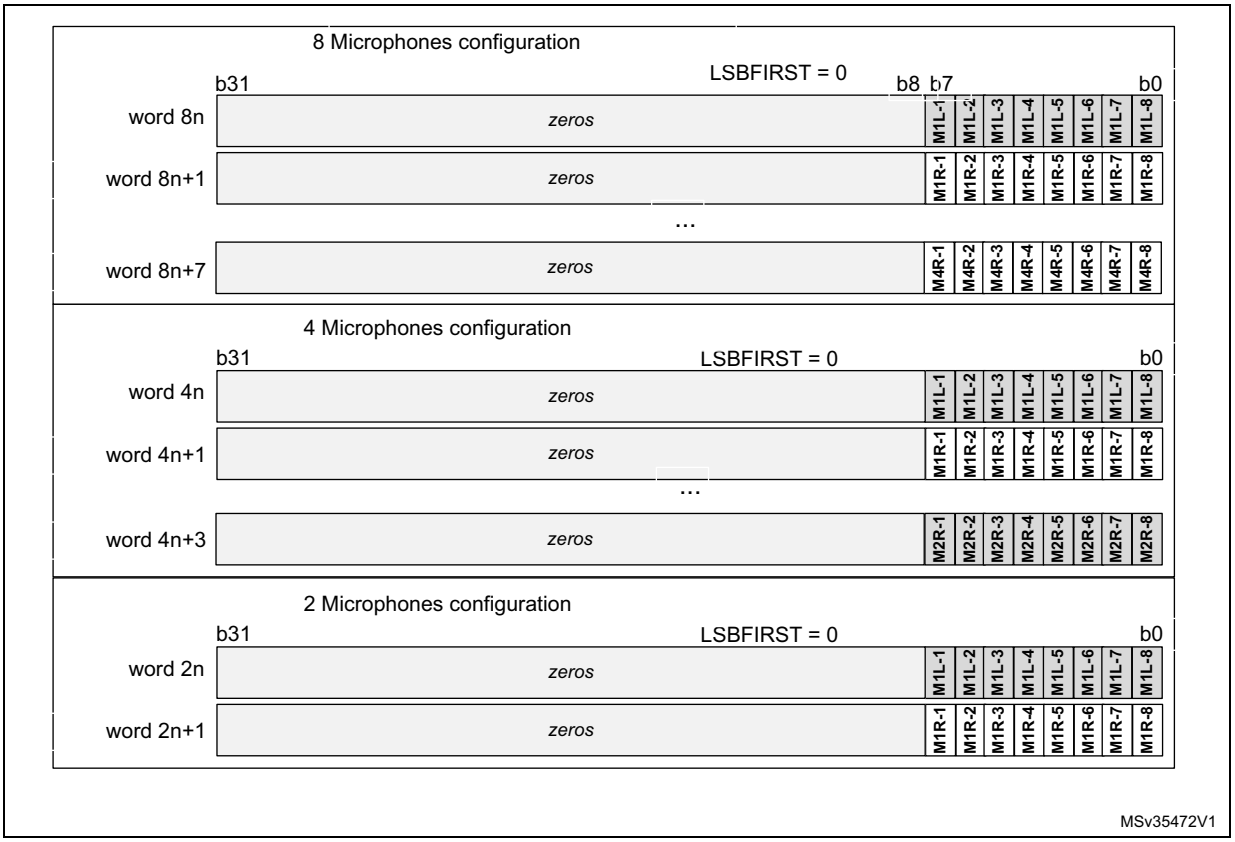


- **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 769*.

  For an 8-microphone configuration, eight consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.

Figure 769. SAI_ADR format in TDM mode, 8-bit slot width



TDM mode configuration for PDM interface

The SAI_A serial audio interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the SAI_A serial audio interface as shown in *Table 586* to ensure a good connection with the PDM interface.

Table 586. SAI_A configuration for TDM mode

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver.
PRTCFG	0b00	Free protocol for TDM.
DS	X	To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See *Table 587*.
LSBFIRST	X	This parameter can be used according to the desired data format.
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode.
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 587*.
FSALL	0	Pulse width is one bit clock cycle.

Table 586. SAI_A configuration for TDM mode (continued)

Bit Fields	Values	Comments
FSDEF	0	FS signal is a start of frame.
FSPOL	1	FS is active high.
FSOFF	0	FS is asserted on the first bit of slot 0.
FBOFF	0	No offset on slot.
SLOTSZ	0	Slot size = data size.
NBSLOT	X	To be adjusted according to the required data format, in accordance with the slot size, and the frame length (FRL and DS). See *Table 587*.
SLOTEN	X	To be adjusted according to NBSLOT.
NODIV	1	No need to generate a master clock MCLK.
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See *Table 587*.

Adjusting the bitstream clock rate

To program the serial audio interface properly, the user application must take into account the settings given in *Table 586*, and follow the sequence below:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2$$

   MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones; see *Section 53.6.18*)

2. Set the frame length (FRL) using the following formula:

$$FRL = (16 \times (MICNBR + 1)) - 1$$

Table 587. TDM frame configuration examples(1)(2)

Microphone sampling rate	Nber of microphones	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 586: SAI_A configuration for TDM mode* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI must be a multiple of the SCK_A frequency, and MCKDIV must be programmed accordingly.
2. The above sai_a_ker_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device*.
3. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMDLY register.

The new delay values become effective after two audio frames.

53.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

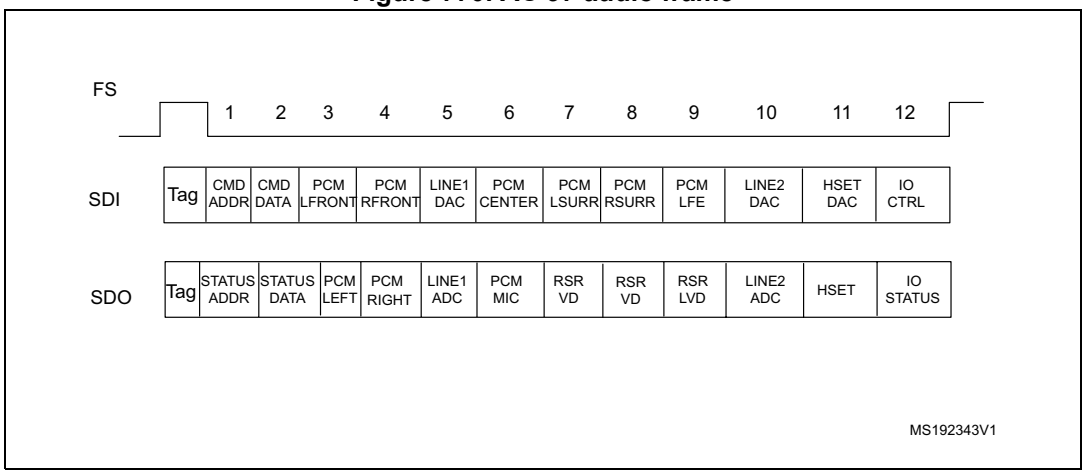
To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 770 shows an AC’97 audio frame structure.

Figure 770. AC’97 audio frame



Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

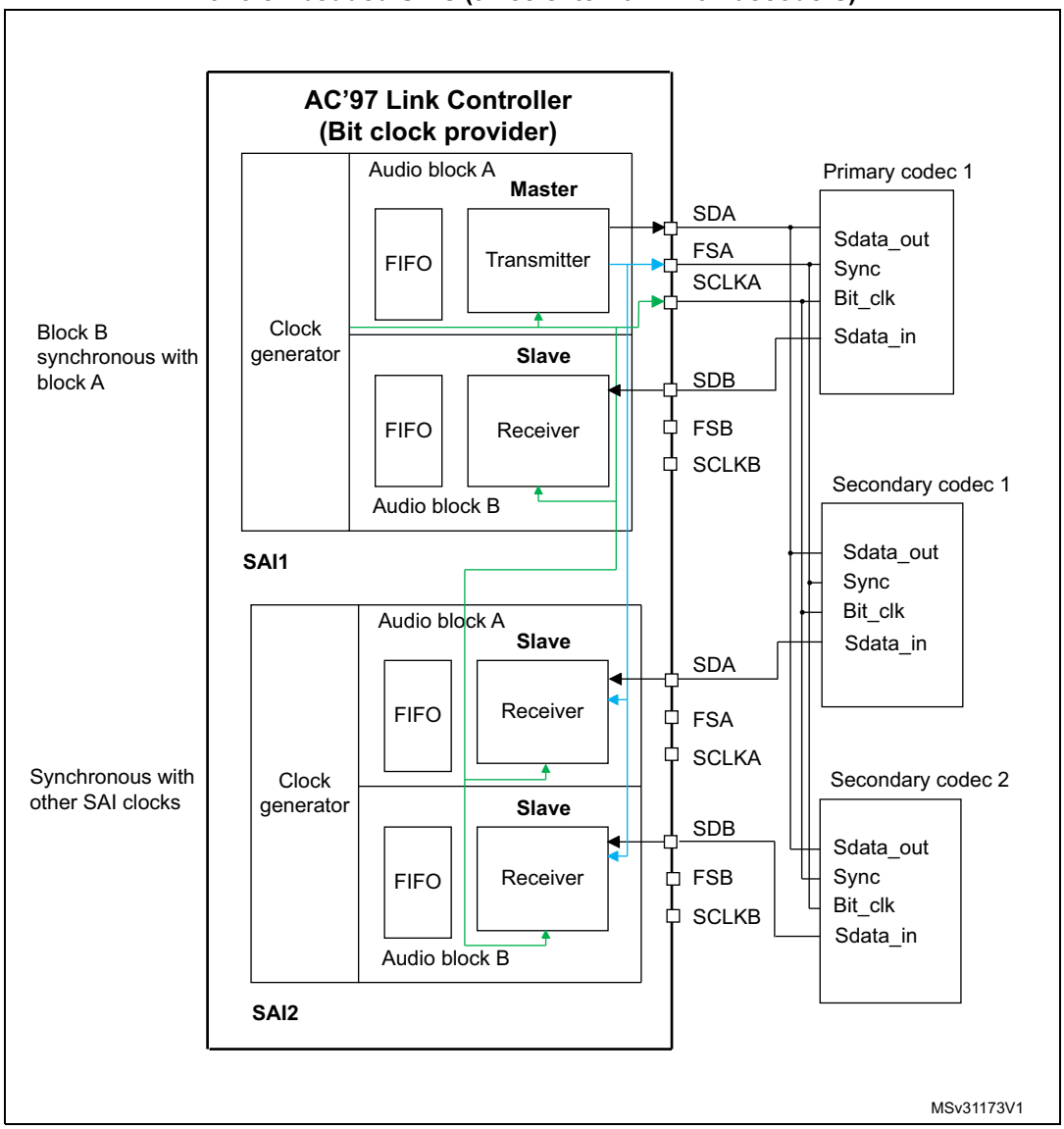
One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) enables the control of three external AC’97 decoders as illustrated in *Figure 771*.

In SAI1, the audio block A must be declared as asynchronous master transmitter, whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.

Figure 771. Example of typical AC'97 configuration on devices featuring at least two embedded SAIs (three external AC'97 decoders)



In receiver mode, the SAI acting as an AC'97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC'97 protocol.

Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in *Section 53.4.8: SAI clock generator* must be used with FRL = 255, to generate the proper frame rate (F_{FS_x}).

53.4.12 SPDIF output

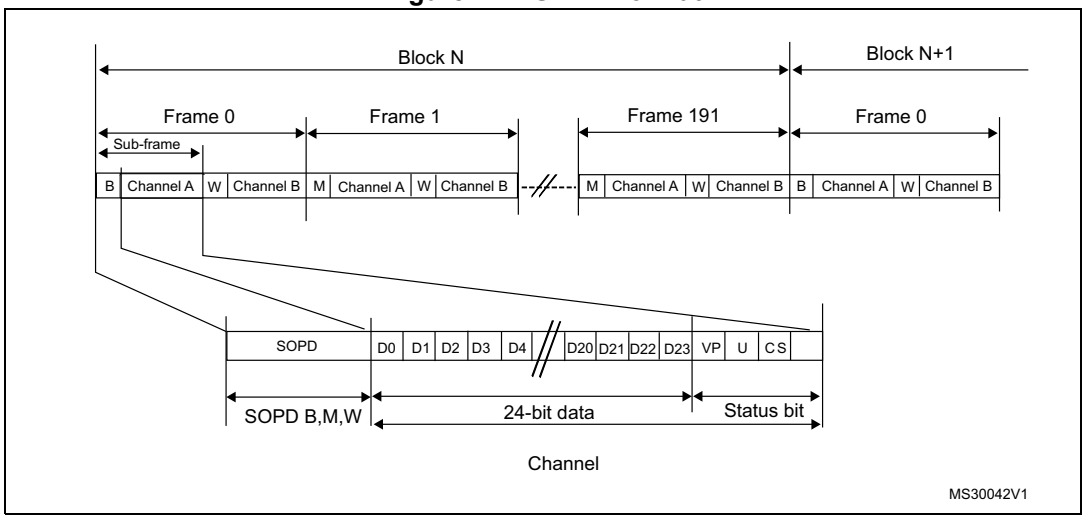
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 772*.

Figure 772. SPDIF format



An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 588*). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 588. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

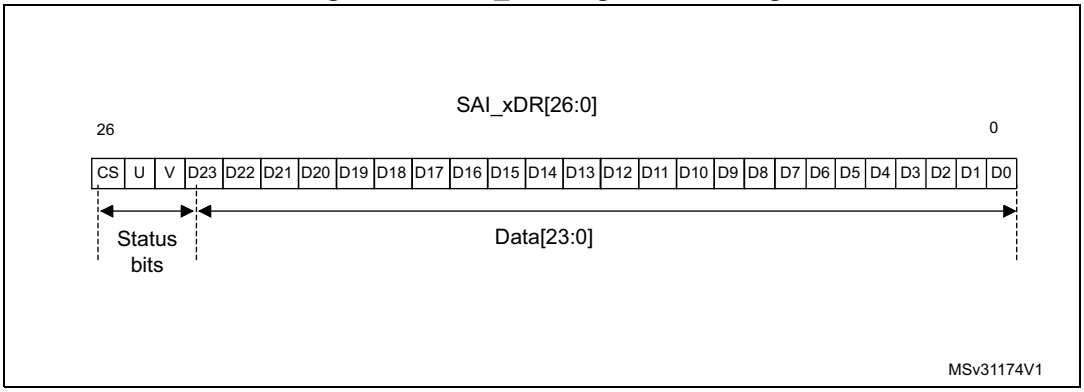
- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 773. SAI_xDR register ordering



Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in *Table 589*.

Table 589. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

Table 590. Audio sampling frequency versus symbol rates

Audio sampling frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_X}) is given by the formula:

$$F_S = \frac{F_{SCK_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.

53.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:

1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in *Figure 774*. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

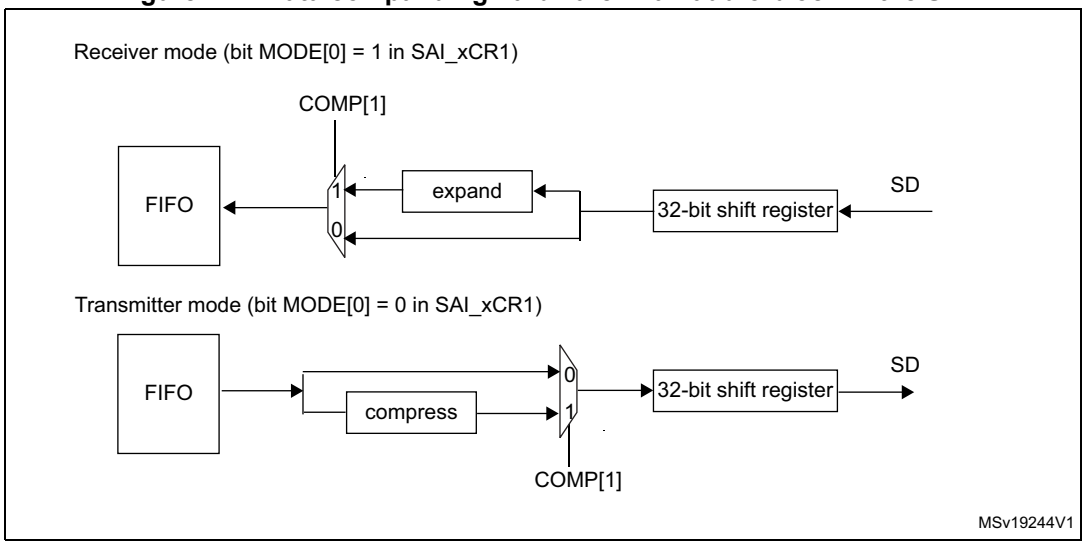
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1's complement or 2's complement representation, depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

Figure 774. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).

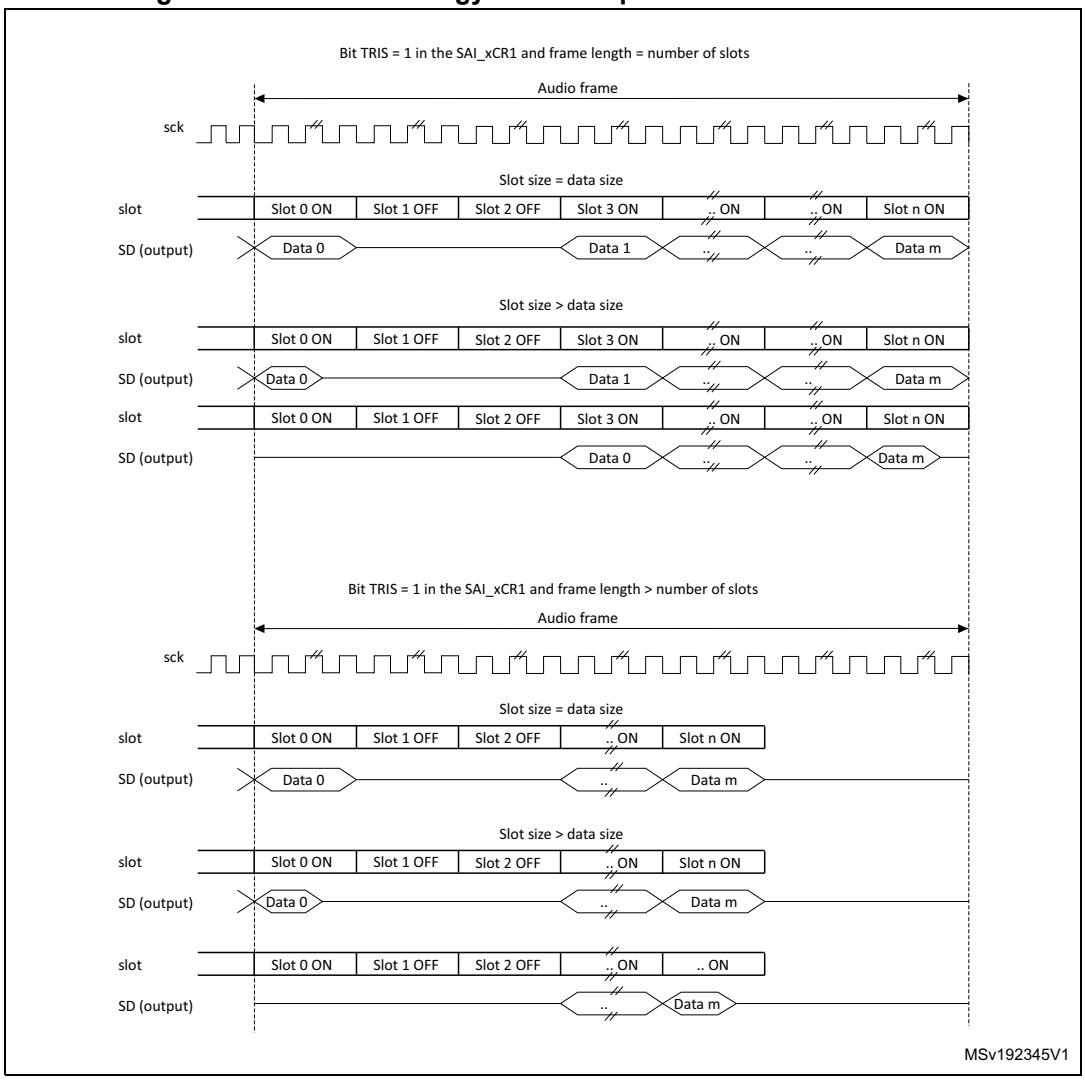
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

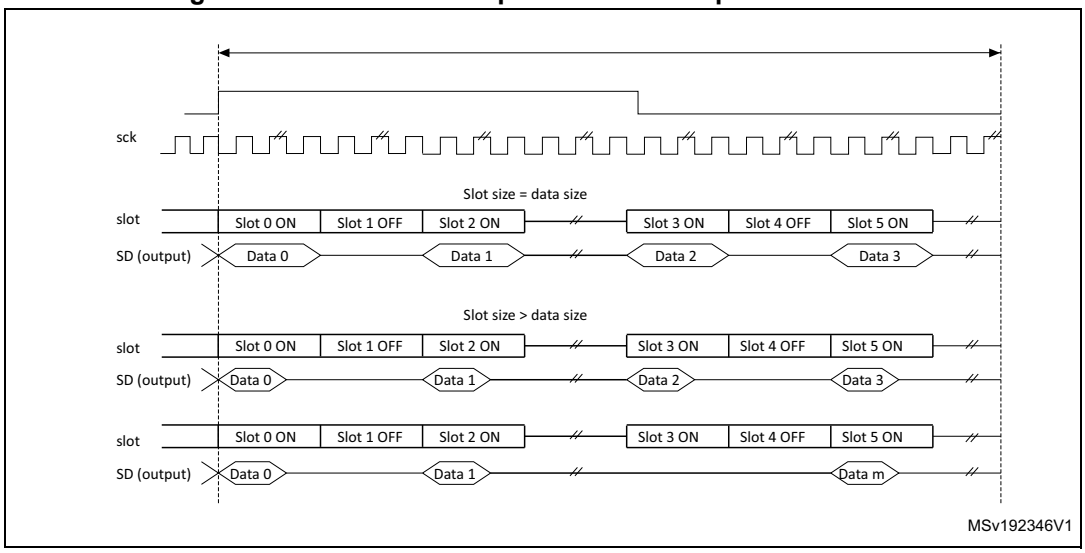
Figure 775 illustrates these behaviors.

Figure 775. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 776* (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 776. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in *Figure 775* and *Figure 776* are replaced by a drive with a value of 0.

53.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC'97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

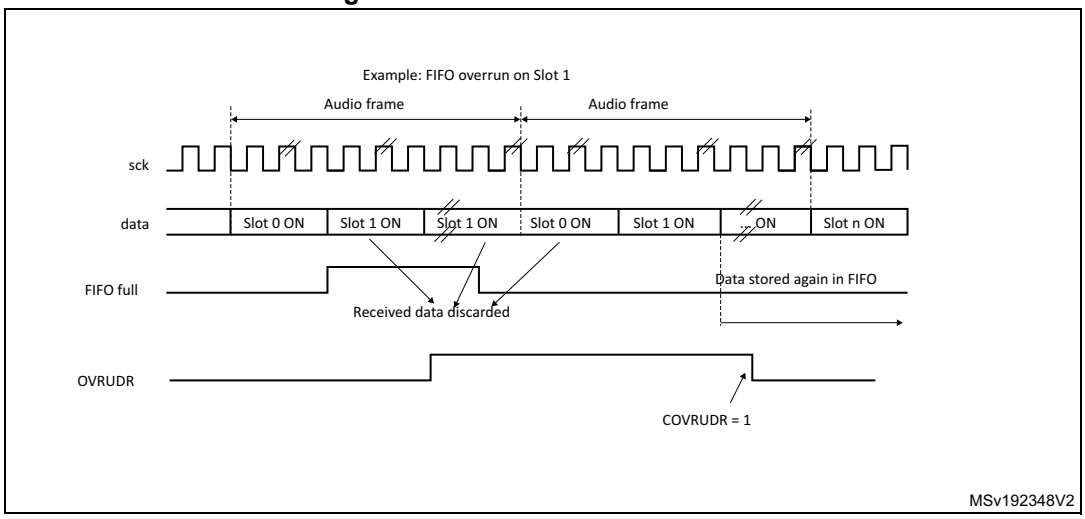
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to *Figure 777*).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.

Figure 777. Overrun detection error



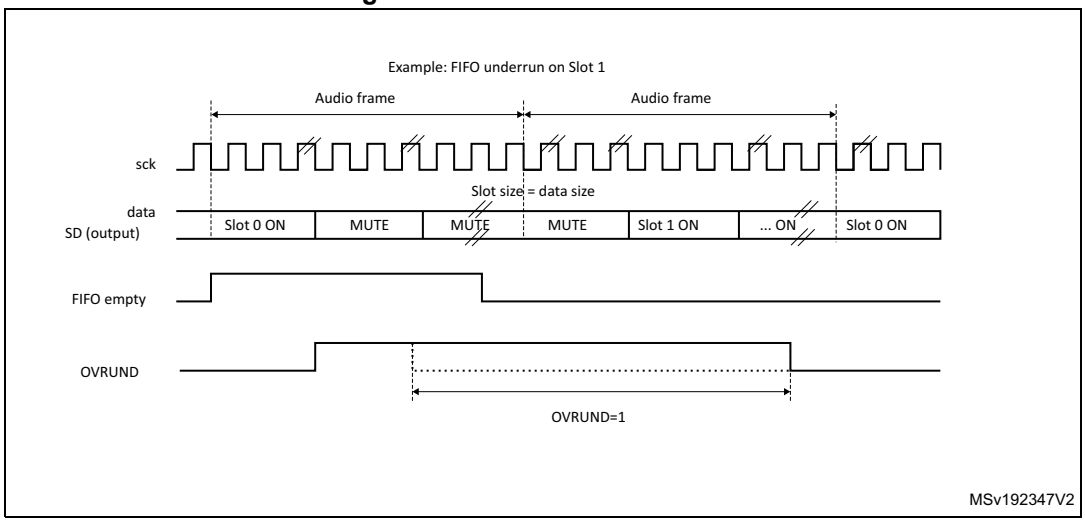
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 778*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 778. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

53.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

53.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 53.4.9: Internal FIFOs*). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

53.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in *Table 591*.

Table 591. SAI interrupt sources

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 53.4.9: Internal FIFOs*
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

53.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

53.6.1 SAI global configuration register (SAI_GCR)

Address offset: 0x00

Reset value: 0x0000 0000

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

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

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs

These bits are set and cleared by software.

00: No synchronization output signals. SYNCOUT[1:0] must be configured as “No synchronization output signals” when audio block is configured as SPDIF

01: Block A used for further synchronization for others SAI

10: Block B used for further synchronization for others SAI

11: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs

These bits are set and cleared by software.

Refer to *Table 583: External synchronization selection* for information on how to program this field.

These bits must be set when both audio blocks (A and B) are disabled.

They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

53.6.2 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 53.4.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

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

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

53.6.3 SAI configuration register 2 (SAI_ACR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

53.6.4 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

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

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

53.6.5 SAI slot register (SAI_ASLOTR)

Address offset: 0x10

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

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

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

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

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

53.6.6 SAI interrupt mask register (SAI_AIM)

Address offset: 0x14

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

53.6.7 SAI status register (SAI_ASR)

Address offset: 0x18

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

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

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 53.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

53.6.8 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

53.6.9 SAI data register (SAI_ADR)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

53.6.10 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, the master clock frequency is calculated according to the formula given in *Section 53.4.8*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

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

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

53.6.11 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1’s complement representation.

1: 2’s complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

53.6.12 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

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

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

53.6.13 SAI slot register (SAI_BSLOTR)

Address offset: 0x30

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

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

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

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

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

0000: Number of slots is 1.

0001: Number of slots is 2.

...

1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

53.6.14 SAI interrupt mask register (SAI_BIM)

Address offset: 0x34

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

53.6.15 SAI status register (SAI_BSR)

Address offset: 0x38

Reset value: 0x0000 0008

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

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

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 53.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

53.6.16 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

53.6.17 SAI data register (SAI_BDR)

Address offset: 0x40

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

53.6.18 SAI PDM control register (SAI_PDMCR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
						rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2

This bit is set and cleared by software.

0: SAI_CK2 clock disabled

1: SAI_CK2 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK2 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1

This bit is set and cleared by software.

0: SAI_CK1 clock disabled

1: SAI_CK1 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK1 might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones

01: Configuration with 4 microphones

10: Configuration with 6 microphones

11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.**

The complete set of data lines might not be available for all SAI instances. Refer to Section 53.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.

Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled

1: PDM interface enabled

53.6.19 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 of T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 53.3: SAI implementation to check if it is available.

53.6.20 SAI register map

Table 592. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	SAI_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
	Reset value																											0	0			0	0
0x04 or 0x24	SAI_xCR1	Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]		Res.	PRTCFG[1:0]		MODE[1:0]		
	Reset value					0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0
0x08 or 0x28	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]							MUTE VAL	MUTE	TRIS	FFLUS	FTH[2:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C or 0x2C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Table 592. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10 or 0x30	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0
0x14 or 0x34	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x18 or 0x38	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	1	0	0	0
0x1C or 0x3C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
	Reset value																										0	0	0		0	0	0
0x20 or 0x40	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																							0	0			0	0				0
0x48	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

54 FD controller area network (FDCAN)

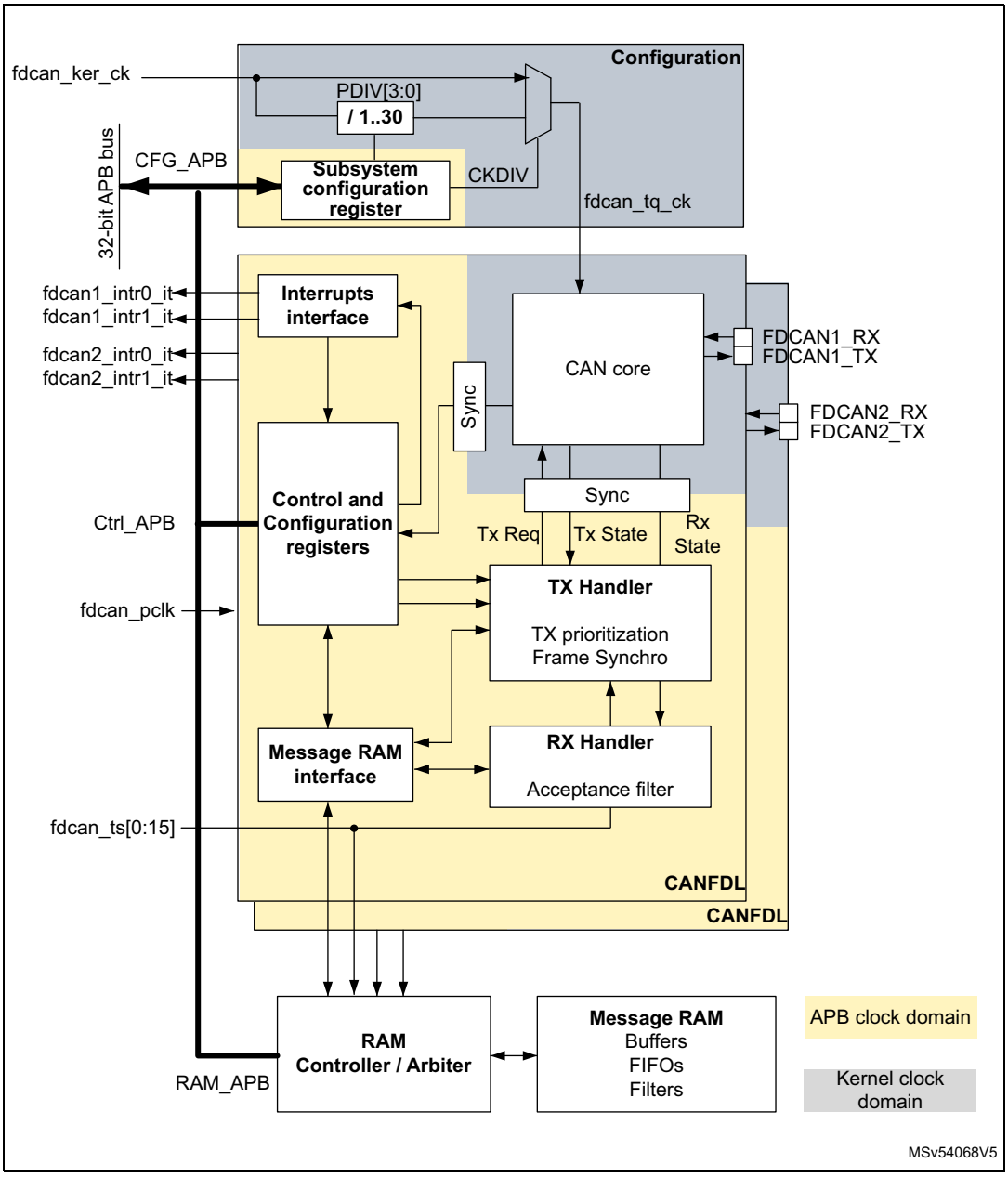
54.1 FDCAN introduction

The controller area network (CAN) subsystem (see *Figure 779*) consists of two CAN modules, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance is used for filtering, transmitting event FIFOs, and receiving and transmitting FIFOs.

Figure 779. CAN subsystem.



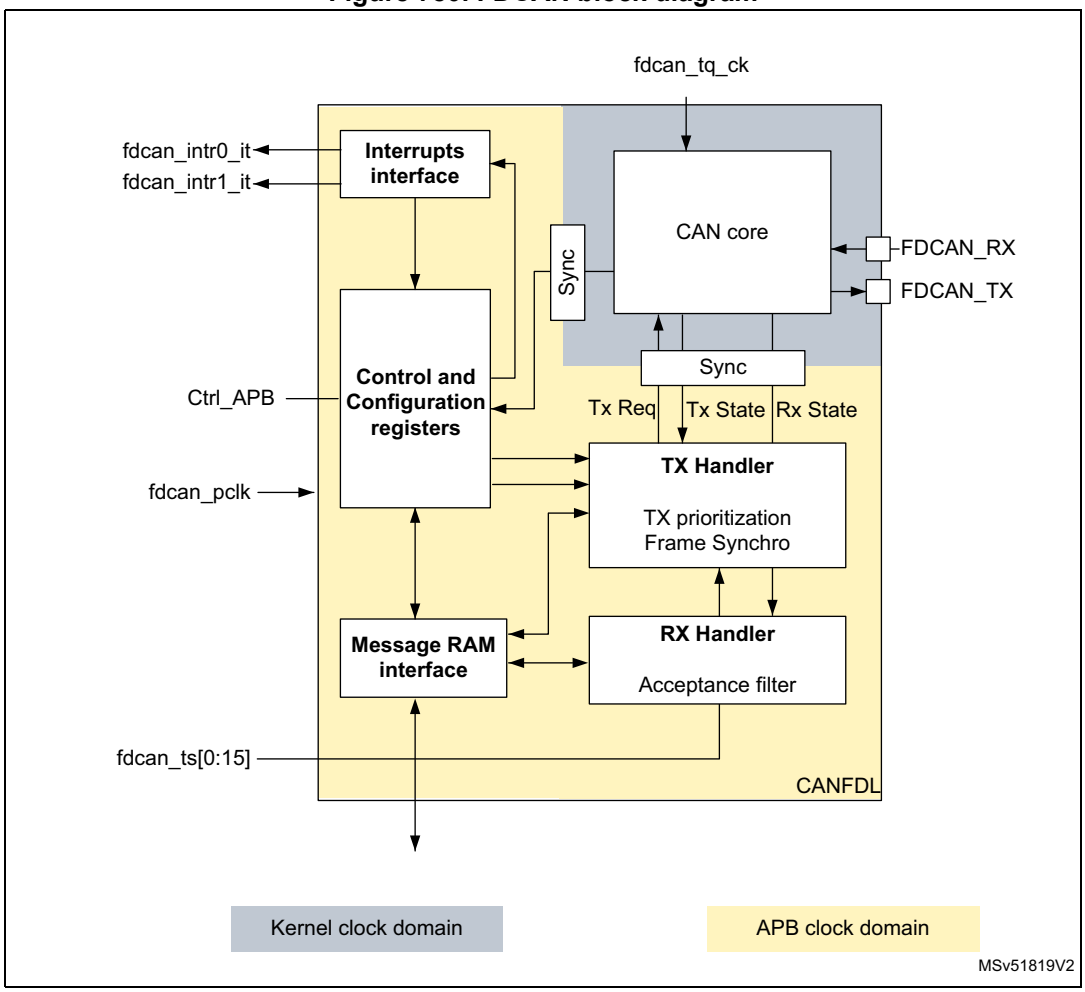
54.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

54.3 FDCAN functional description

54.3.1 FDCAN block diagram

Figure 780. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.

By programming the EINT0 and EINT1 bits of the FDCAN_ILE register, the interrupt lines can be independently enabled or disabled.

CAN core

The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.

Tx handler

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

Rx handler

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

APB interface

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

Message RAM interface

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbiter.

54.3.2 FDCAN pins and internal signals

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 593*, *Table 594*, and *Figure 779*.

Table 593. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan*n*_intr0_it	Digital output	FDCAN interrupt0
fdcan*n*_intr1_it		FDCAN interrupt1
fdcan_ts[0:15]	-	External timestamp vector
APB interface	Digital input/output	Single APP with multiple psel for configuration, control and RAM access

Table 594. CAN subsystem I/O pins

Name	Type	Description
FDCAN*n*_RX	Digital input	FDCAN receive pin
FDCAN*n*_TX	Digital output	FDCAN transmit pin

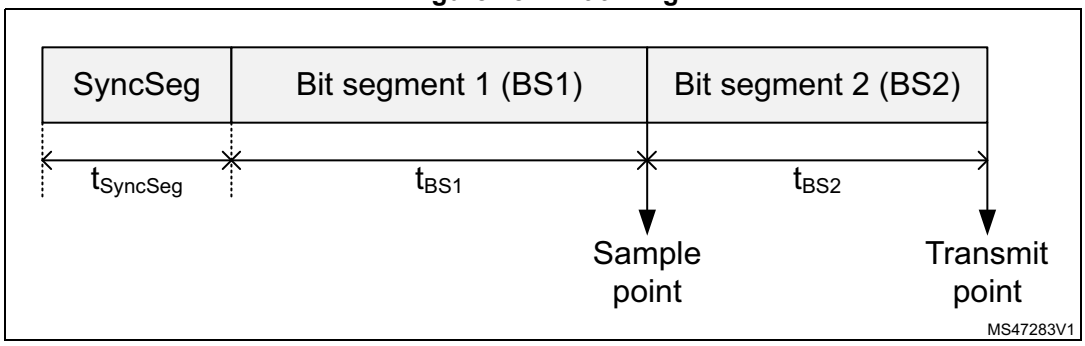
54.3.3 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in *Figure 781*, this operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum ($1 \times t_q$).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable from 1 to 32 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but can also be automatically shortened to compensate for negative phase drifts.

Figure 781. Bit timing



The baud rate is the inverse of the bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components (see *Figure 781*):

bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$

Where:

– For the nominal bit time

$t_q = (NBRP[8:0] + 1) \times t_{fdcan_tq_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (NTSEG1[7:0] + 1)$

$t_{BS2} = t_q \times (NTSEG2[6:0] + 1)$

Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN_NBTP register.

– For the data bit time

$t_q = (DBRP[4:0] + 1) \times t_{fdcan_tq_clk}$

$t_{SyncSeg} = 1 \times t_q$

$t_{BS1} = t_q \times (DTSEG1[4:0] + 1)$

$t_{BS2} = t_q \times (DTSEG2[3:0] + 1)$

Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN_DBTP and FDCAN_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA_CCCR register are set.

The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock ($f_{dcan_tq_ck} < f_{dcan_pclk}$).

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

54.3.4 Operating modes

Configuration

Access to peripheral version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

Software initialization

Software initialization is started by setting the INIT bit of the FDCAN_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN_CCCR register are both set.

The CCE bit of the FDCAN_CCCR register can only be set/cleared while the INIT bit of FDCAN_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.

The following registers are reset when the CCE bit of the FDCAN_CCCR register is set:

- FDCAN_HPMS: High priority message status
- FDCAN_RXF0S: Rx FIFO 0 status
- FDCAN_RXF1S: Rx FIFO 1 status
- FDCAN_TXFQS: Tx FIFO/queue status
- FDCAN_TXBRP: Tx buffer request pending
- FDCAN_TXBTO: Tx buffer transmission occurred
- FDCAN_TXBCF: Tx buffer cancellation finished
- FDCAN_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN_TOCC register when the CCE bit of the FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN_TXBAR: Tx buffer add request
- FDCAN_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN_CCCR can only be set/cleared while the INIT and CCE bits are both set.

Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

- Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes.
- Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame.

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN_PSR) to integrating (ACT[1:0] = 00 in FDCAN_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRS bits set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 595*.

Table 595. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN_DBTP register. In the following

FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (t_q), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mt_q (minimum time quantum, one period of fdcan_tq_ck clock).

The TDCV[6:0] bitfield of the FDCAN_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN_CCCR. It is

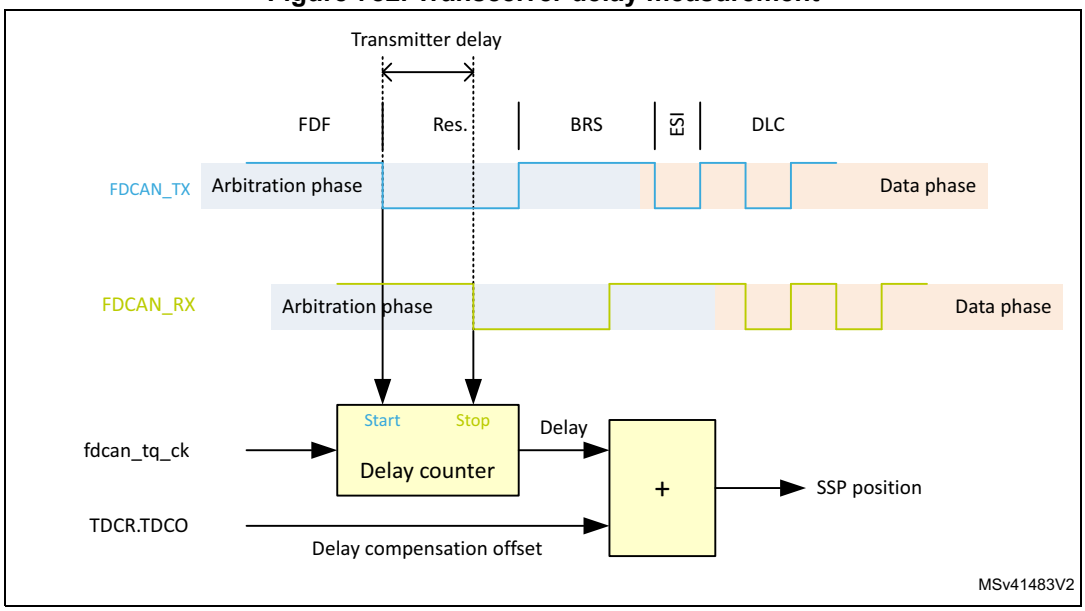
updated at each transmission of an FD frame while the TDC bit of the FDCAN_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to 127 × mt_q. If the sum exceeds this value, the maximum value (127 × mt_q) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mt_q.

Figure 782. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

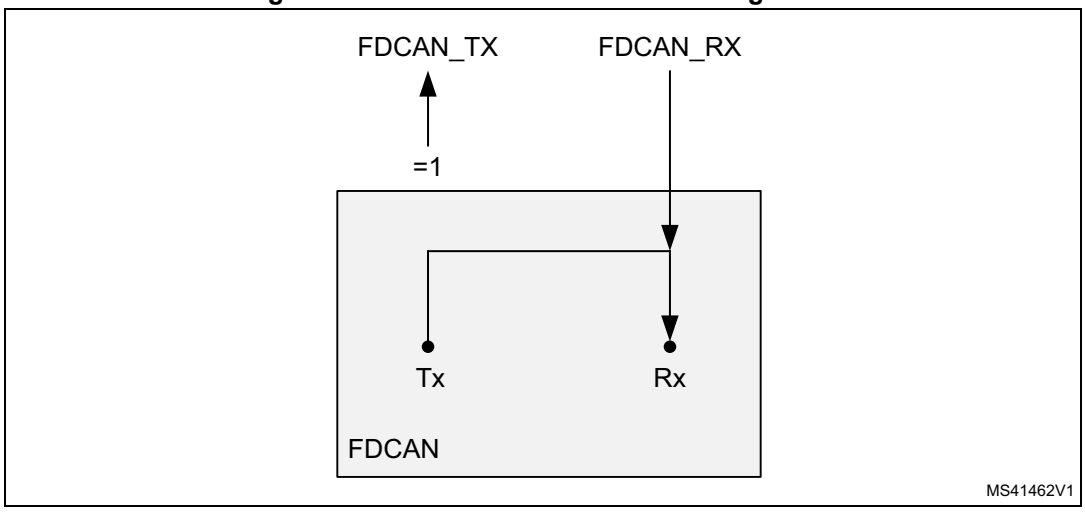
Note: The restricted operation mode must not be combined with the loop-back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 783* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 783. Pin control in bus monitoring mode



Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN_CCCR register.

Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN_TXBCF register.
- Successful transmission in spite of cancellation
  - The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.
- Arbitration loss or frame transmission disturbed
  - The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power-down (Sleep mode)

Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

Test modes

To enable write access to *FDCAN test register (FDCAN_TEST)*, the TEST bit of the FDCAN_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming the TX[1:0] bitfield of the FDCAN_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from the RX bit of FDCAN_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via RX.

Note: Test modes must be used for production tests or self-test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop-back mode

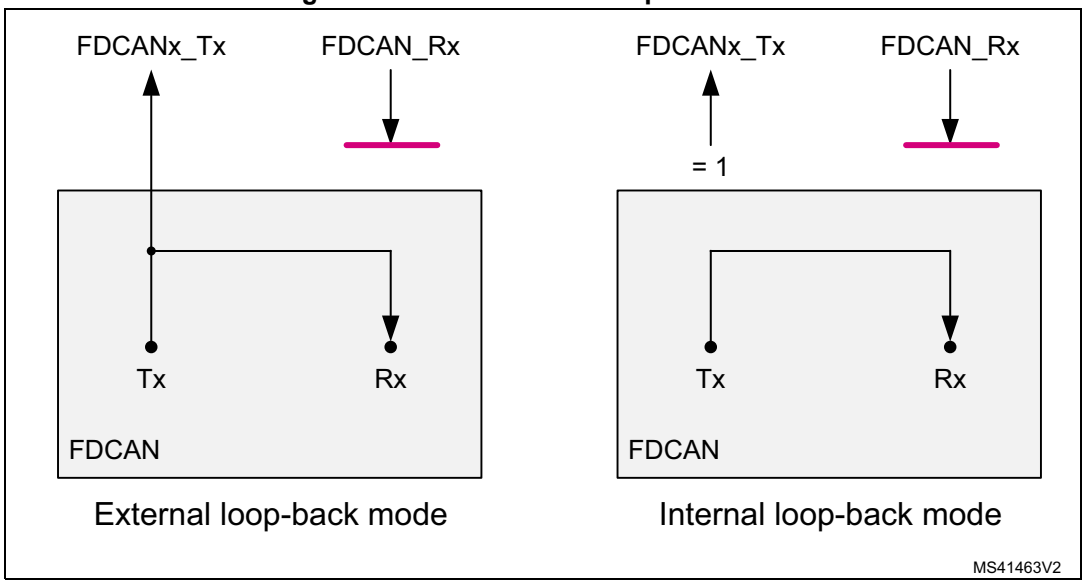
The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 784* shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop-back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN_TEST and the MON bit of FDCAN_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. *Figure 784* shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop-back mode.

Figure 784. Pin control in loop-back mode



Timestamp generation

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN_TSCV register. A write access to TSCV15:0] resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming TSS[1:0] of FDCAN_TSCC, a 16-bit timestamp can be used.

Debug mode behavior

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN_TOCV. The timeout counter can only be started while the INIT bit of FDCAN_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the TOO interrupt flag is set in the FDCAN_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

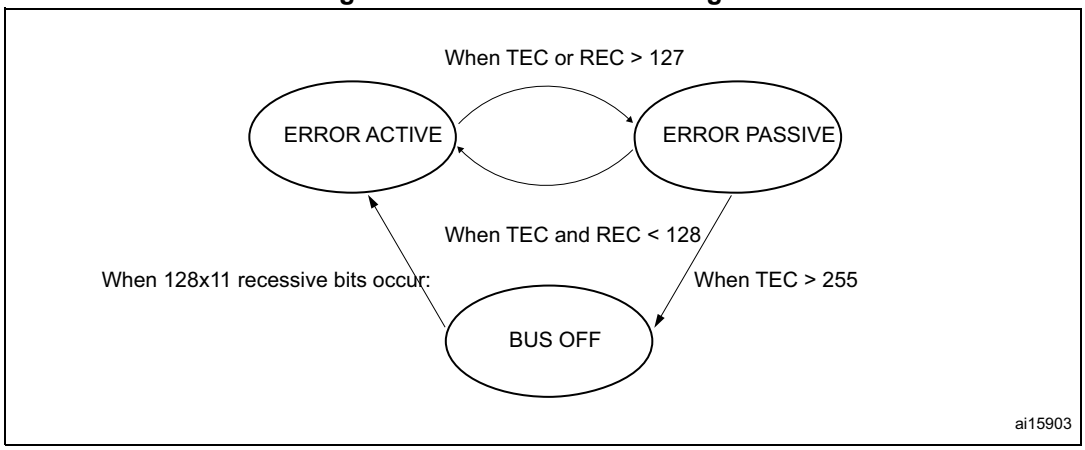
Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.

54.3.5 Error management

As described in the CAN protocol, the error management is handled entirely by hardware using the transmit error counter (the TEC[7:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*) and the receive error counter (the REC[6:0] bitfield of the *FDCAN error counter register (FDCAN_ECR)*). These values are incremented or decremented according to the error condition. For detailed information on TEC[7:0] and REC[6:0] management, refer to the CAN standard. Both values can be read by software to determine the stability of the network.

The bus-off state is reached when TEC[7:0] is greater than 255. This state is also indicated by the BO flag of the *FDCAN protocol status register (FDCAN_PSR)*. In bus-off state, the CAN is no longer able to transmit and receive messages. It has to wait at least for the duration of the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on FDCAN_RX input).

Figure 785. CAN error state diagram

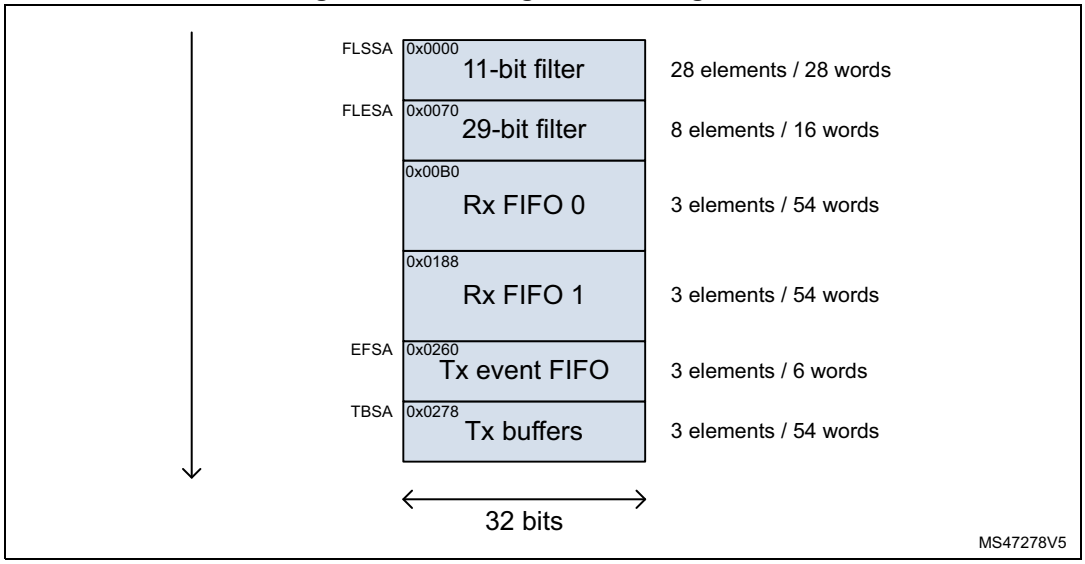


Note: In initialization mode, the CAN does not monitor the FDCAN_RX *signal, and therefore cannot complete the recovery sequence. To recover from an error state, the CAN must operate in normal mode.*

54.3.6 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in *Figure 786*.

Figure 786. Message RAM configuration



When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:

- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in *Figure 786*)
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN_IR
- Set the high priority message interrupt flag HPM in FDCAN_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**

  The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT[1:0] = 00: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- EFT[1:0] = 11: the extended ID AND mask (XIDAM) is not used for range filtering.

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

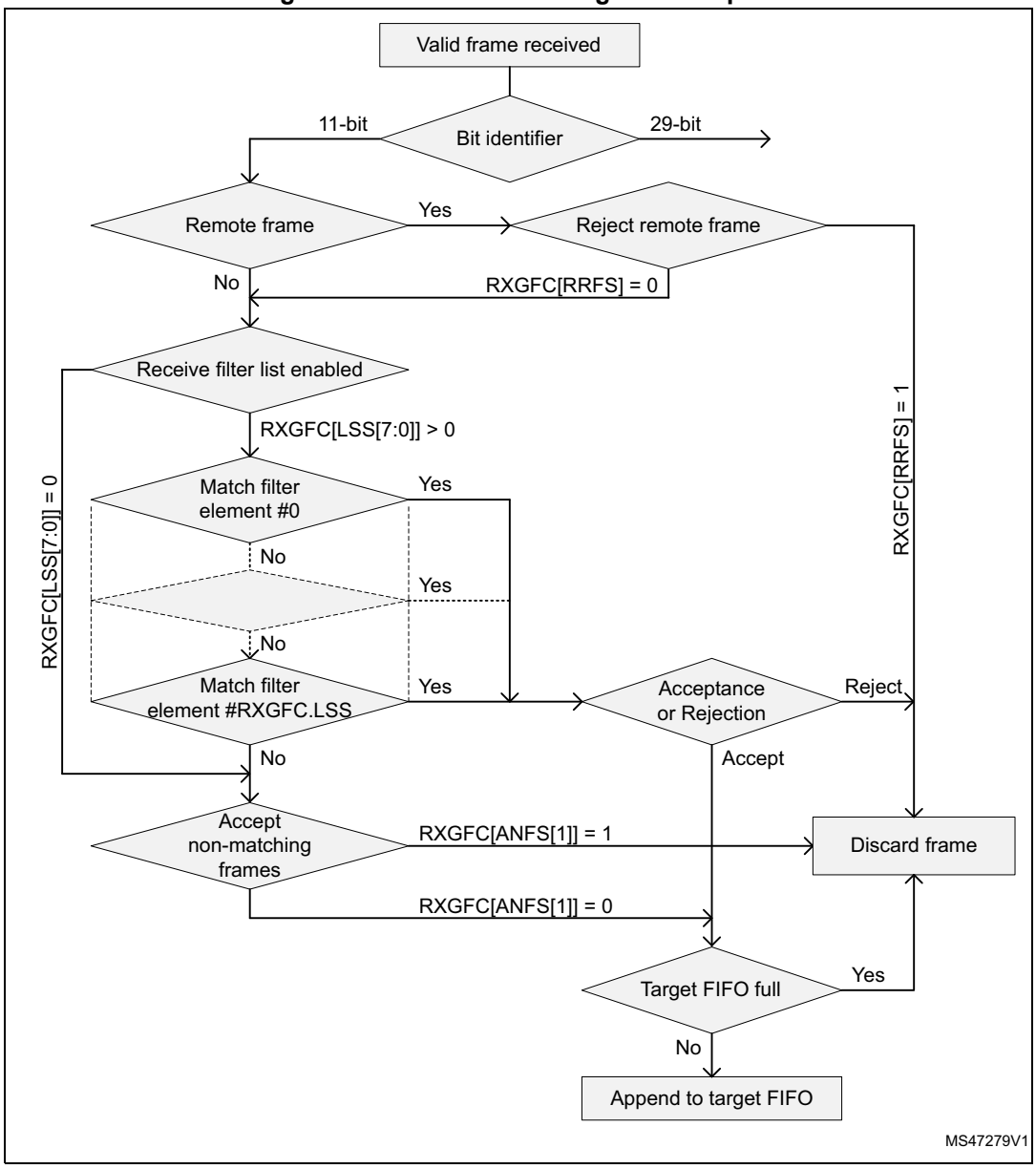
In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 787 shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in *Section 54.3.11*.

The standard message filtering is controlled by the FDCAN_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 787. Standard message ID filter path

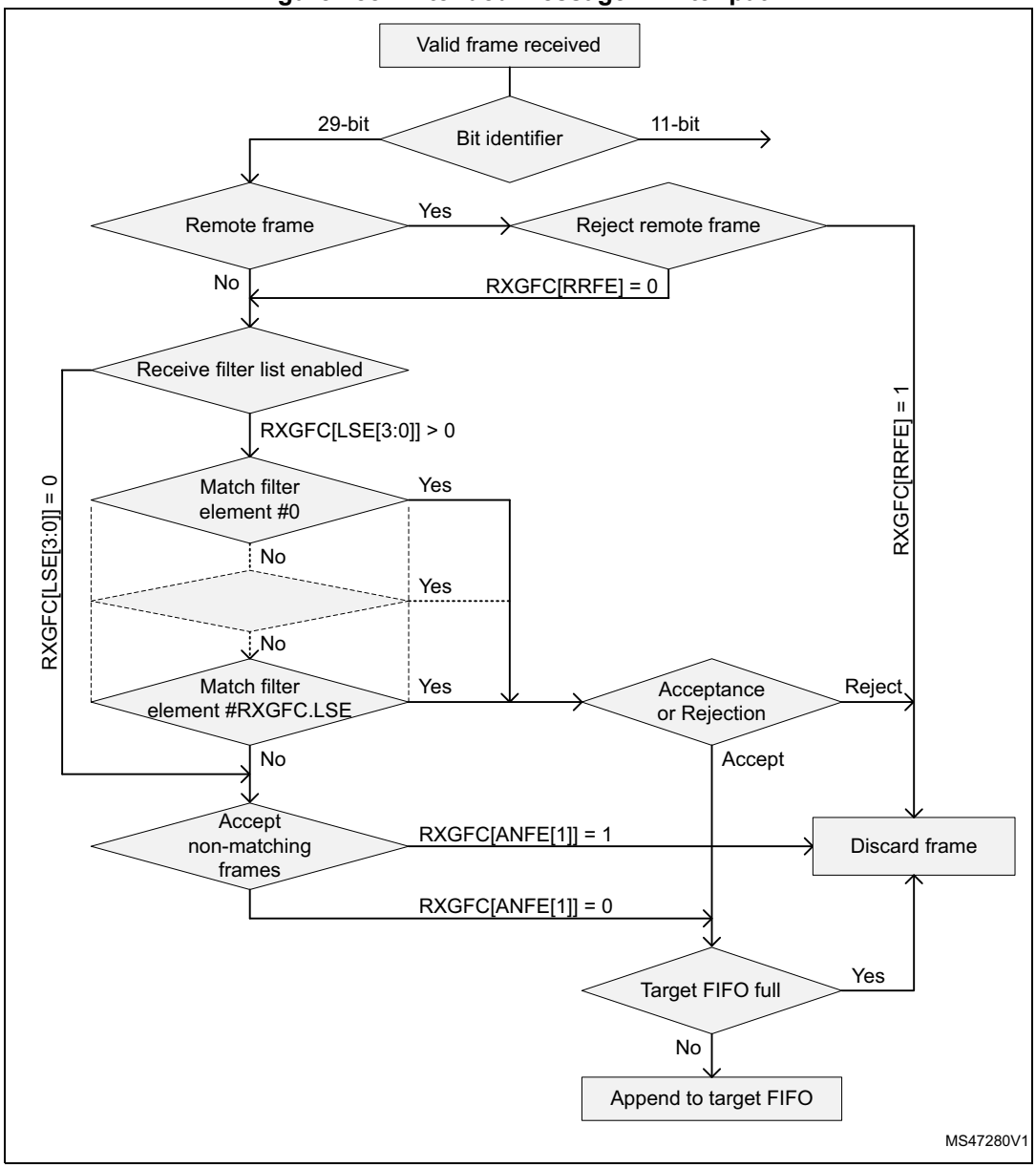


Extended message ID filtering

Figure 788 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in *Section 54.3.12*.

The extended message filtering is controlled by the FDCAN_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 788. Extended message ID filter path



The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx FIFO element is described in *Section 54.3.8*.

When an Rx FIFO full condition is signaled by RFnF in FDCAN_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGI of FDCAN_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN_IR.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGI of FDCAN_RXFnS) is signaled by FnF = 1 in FDCAN_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN_RXFnS).

Tx handling

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. the Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 596. Possible configurations for frame transmission

CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bit rate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN_TXBRP) is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN_CCCR).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Tx FIFO

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCA_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. The Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding 18 times the put index TFQPI[1:0] (0 … 2) to the Tx buffer start address TBSA.

Tx queue

Tx queue operation is configured by setting the TFQM of the FDCAN_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

The application can use the FDCAN_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding 18 times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN _TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in *Tx FIFO*.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.

54.3.7 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see *Section 54.4.23* and *Section 54.4.25*). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

54.3.8 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in *Table 597*. The description is provided in *Table 598*.

Table 597. Rx FIFO element

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R0	ESI	XTD	RTR	ID[28:0]																												
R1	ANMF	FIDX[6:0]							Res.				FDF	BRS	DLC[3:0]		RXTS[15:0]															
R2	DB3[7:0]								DB2[7:0]								DB1[7:0]								D[7:0]							
R3	DB7[7:0]								DB6[7:0]								DB5[7:0]								DB4[7:0]							
⋮	⋮								⋮								⋮															
Rn	DBm[7:0]								DBm-1[7:0]								DBm-2[7:0]								DBm-3[7:0]							

The element size configured for storage of CAN FD messages is set to 64-byte data field.

Table 598. Rx FIFO element description

Field	Description
R0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
R0 Bit 30 XTD	Extended identifier Signals to the host whether the received frame has a standard or extended identifier. – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 598. Rx FIFO element description (continued)

Field	Description
R0 Bit 29 RTR	Remote transmission request Signals to the host whether the received frame is a data frame or a remote frame. – 0: Received frame is a data frame – 1: Received frame is a remote frame
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC. – 0: Received frame matching filter index FIDX – 1: Received frame did not match any Rx filter element
R1 Bits 30:24 FIDX[6:0]	Filter index 0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1). Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.
R1 Bit 21 FDF	FD format – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit rate switch – 0: Frame received without bit rate switching – 1: Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data length code – 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9-15: Classic CAN: received frame has 8 data bytes – 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.
R2 Bits 31:24 DB3[7:0]	Data byte 3
R2 Bits 23:16 DB2[7:0]	Data byte 2
R2 Bits 15:8 DB1[7:0]	Data byte 1
R2 Bits 7:0 D[7:0]	Data byte 0
R3 Bits 31:24 DB7[7:0]	Data byte 7
R3 Bits 23:16 DB6[7:0]	Data byte 6

Table 598. Rx FIFO element description (continued)

Field	Description
R3 Bits 15:8 DB5[7:0]	Data byte 5
R3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮
Rn Bits 31:24 DBm[7:0]	Data byte m
Rn Bits 23:16 DBm-1[7:0]	Data byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data byte m-3

54.3.9 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration TFQM bit of the FDCAN_TXBC register. The element size is configured for storage of CAN FD messages with up to 64-byte data.

Table 599. Tx buffer and FIFO element

Bit	31 24	23 16	15 8	7 0
T0	ESI | XTD | RTR | ID[28:0]			
T1	MM[7:0]	EFC | Res. | FDF | BRS | DLC[3:0]	Res.	
T2	DB3[7:0]	DB2[7:0]	DB1[7:0]	D[7:0]
T3	DB7[7:0]	DB6[7:0]	DB5[7:0]	DB4[7:0]
⋮	⋮	⋮	⋮	
Tn	DBm[7:0]	DBm-1[7:0]	DBm-2[7:0]	DBm-3[7:0]

Table 600. Tx buffer element description

Field	Description
T0 Bit 31 ESI(1)	Error state indicator – 0: ESI bit in CAN FD format depends only on error passive flag – 1: ESI bit in CAN FD format transmitted recessive
T0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier

Table 600. Tx buffer element description (continued)

Field	Description
T0 Bit 29 RTR(2)	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO control – 0: Do not store Tx events – 1: Store Tx events
T1 Bit 21 FDF	FD format – 0: Frame transmitted in classic CAN format – 1: Frame transmitted in CAN FD format
T1 Bit 20 BRS(3)	Bit rate switching – 0: CAN FD frames transmitted without bit rate switching – 1: CAN FD frames transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes – 9 - 15: Classic CAN: received frame has 8 data bytes – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data byte 3
T2 Bits 23:16 DB2[7:0]	Data byte 2
T2 Bits 15:8 DB1[7:0]	Data byte 1
T2 Bits 7:0 D[7:0]	Data byte 0
T3 Bits 31:24 DB7[7:0]	Data byte 7
T3 Bits 23:16 DB6[7:0]	Data byte 6
T3 Bits 15:8 DB5[7:0]	Data byte 5
T3 Bits 7:0 DB4[7:0]	Data byte 4
⋮	⋮

Table 600. Tx buffer element description (continued)

Field	Description
Tn Bits 31:24 DBm[7:0]	Data byte m
Tn Bits 23:16 DBm-1[7:0]	Data byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node can optionally transmit the ESI bit recessive, but an error passive node always transmits the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if the transmission in CAN FD format is enabled by the FDOE bit of the FDCAN_CCCR.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled by setting the FDOE bit of the FDCAN_CCCR. Bit BRS is only evaluated when in addition BRSE bit is set in FDCAN_CCCR.

54.3.10 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN_TXEFS register.

Table 601. Tx event FIFO element

Bit	31 24	23 16	15 8	7 0
E0	ESI, XTD, RTR, ID[28:0]			
E1	MM[7:0]	ET[1:0], EDL, BRS, DLC[3:0]	TXTS[15:0]	

Table 602. Tx event FIFO element description

Field	Description
E0 Bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 Bit 30 XTD	Extended identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 Bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].

Table 602. Tx event FIFO element description (continued)

Field	Description
E1 Bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.
E1 Bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 Bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit rate switching – 0: Frame transmitted without bit rate switching – 1: Frame transmitted with bit rate switching
T1 Bits 19:16 DLC[3:0]	Data length code 0 - 8: Frame with 0-8 data bytes transmitted 9 - 15: Frame with 8 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC.

54.3.11 FDCAN standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 … 27).

Table 603. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 604. Standard message ID filter element field description

Field	Description
Bit 31:30 SFT[1:0](1)	Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]	Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR.HPM and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
Bits 26:16 SFID1[10:0]	Standard filter ID 1 First ID of standard ID filter element.
Bits 10:0 SFID2[10:0]	Standard filter ID 2 Second ID of standard ID filter element.

1. With SFT[1:0] = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC[2:0] = 000).

Note: In case a reserved value is configured, the filter element is considered disabled.

54.3.12 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 … 7).

Table 605. Extended message ID filter element

Bit	31		24	23 16	15 8	7 0
F0	EFEC[2:0]		EFID1[28:0]			
F1	EFT[1:0]	Res.	EFID2[28:0]			

Table 606. Extended message ID filter element field description

Field	Description
F0 Bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Not used
F0 Bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism.
F1 Bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied
F1 Bit 29	Not used
F1 Bits 28:0 EFID2[28:0]	Extended filter ID 2 Second ID of extended ID filter element.

54.4 FDCAN registers

54.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

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

Bits 31:28 **REL[3:0]**: 3

Bits 27:24 **STEP[3:0]**: 2

Bits 23:20 **SUBSTEP[3:0]**: 1

Bits 19:16 **YEAR[3:0]**: 4

Bits 15:8 **MON[7:0]**: 12

Bits 7:0 **DAY[7:0]**: 18

54.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

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

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endianness issue.

54.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods: t_q = (DBRP[4:0] + 1) FDCAN clock periods.

DTSEG1[4:0] is the sum of PROP_SEG and PHASE_SEG1. DTSEG2[3:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [DTSEG1[4:0] + DTSEG2[3:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bit rate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values for the baud rate prescaler are 0 to 31. The hardware interpreters this value as the value programmed plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is t_{BS1} = (DTSEG1[4:0] + 1) × t_q.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, i.e. t_{BS2} = (DTSEG2[3:0] + 1) × t_q.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1: t_{SJW} = (DSJW[3:0] + 1) × t_q.

Note: With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s.

The data phase bit rate must be higher than or equal to the nominal bit rate.

54.4.4 FDCAN test register (FDCAN_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

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

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

Bit 7 **RX**: Receive pin

This bit is used to monitor the actual value of FDCANx_RX. It is synchronized with the FDCANx_RX pin, so it is set after reset if the FDCAN is connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop-back mode

0: Reset value, loop-back mode is disabled

1: Loop-back mode is enabled (see *Power-down (Sleep mode)*)

Bits 3:0 Reserved, must be kept at reset value.

54.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN_IR register. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00, the counter is disabled.
This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

54.4.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non-ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP:** Transmit pause enable
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant t_q required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bit rate switching
0: Bit rate switching for transmissions disabled
1: Bit rate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, FDCAN_TEST holds reset values
1: Test mode, write access to FDCAN_TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN can be set in power-down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or over.load frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN_CCCR).

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization started

Note: Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

54.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR register are both set. The CAN bit time can be programed in the range of 4 to 81 × t_q. The CAN time quantum can be programmed in the range of 1 to 1024 FDCAN kernel clock periods:
t_q = (BRP + 1) × FDCAN clock period fdcan_ker_ck.

NTSEG1[7:0] is the sum of PROP_SEG and PHASE_SEG1. NTSEG2[6:0] is PHASE_SEG2. Therefore, the length of the bit time is
(programmed values) × [NTSEG1[7:0] + NTSEG2[6:0] + 3] × t_q or
(functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that the used value is the one programmed incremented by one.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 24:16 **NBRP[8:0]**: Bit rate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quantum. Valid values are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

This bitfield is write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used.

Note: With a CAN kernel clock of 48 MHz, the reset value of 0x0600 0A03 configures the FDCAN for a bit rate of 1.5 Mbit/s.

54.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16].

The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).

This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

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

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value (tim3_cnt[0:15])

11: Same as 00.

These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

54.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When TSS[1:0] = 01 in FDCAN_TSCC, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of TCP[3:0] in FDCAN_TSCC. A wrap around sets the TSW interrupt flag in FDCAN_IR. Write access resets the counter to 0.
When TSS[1:0] = 10, TSC[15:0] reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN_TSCV.

54.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

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

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.
00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1
This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable

0: Timeout counter disabled
1: Timeout counter enabled
This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

For more details, see *Timeout counter*.

54.4.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN_TOCC.

54.4.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN_IR.

Access type is rc_r: cleared on read.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128.

1: The receive error counter has reached the error passive level of 128.

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When the ASM bit of the FDCAN_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented.

54.4.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs	rs	rs	r	r	r	r	r	rs	rs	rs

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

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCO[6:0] in FDCAN_TDCR. The SSP position is, in the data phase, the number of minimum time quanta (mt_q) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 × mt_q.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access

1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.

0: Since this bit was cleared by the CPU, no FDCAN message has been received.

1: Message in FDCAN format with EDL flag set has been received.

Access type is rc_r: cleared on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its BRS flag set.

1: Last received FDCAN message had its BRS flag set.

Access type is rc_r: cleared on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.

0: Last received FDCAN message did not have its ESI flag set.

1: Last received FDCAN message had its ESI flag set.

Access type is rc_r: cleared on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.

Access type is rs: set on read.

Bit 7 **BO**: Bus-off status

0: The FDCAN is not in bus-off state.

1: The FDCAN is in bus-off state.

Bit 6 **EW**: Warning status

0: Both error counters are below the error-warning limit of 96.

1: At least one of error counter has reached the error-warning limit of 96.

Bit 5 **EP**: Error passive

0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.

1: The FDCAN is in the error-passive state.

Bits 4:3 **ACT[1:0]**: Activity

Monitors the module's CAN communication state.

00: Synchronizing: node is synchronizing on CAN communication.

01: Idle: node is neither receiver nor transmitter.

10: Receiver: node is operating as receiver.

11: Transmitter: node is operating as transmitter.

Bits 2:0 **LEC[2:0]:** Last error code

LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.

000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.

001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

010: Form error. A fixed format part of a received frame has the wrong format.

011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.

100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.

101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.

111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

Access type is rs: set on read.

Note: When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.

The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN_PSR, enabling the CPU to check up whether the CAN bus is

stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The REC[6:0] bitfield of the FDCAN_ECR register is used to count these sequences.

54.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset
Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 × mt_q.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length
Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

54.4.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN_IE controls whether an interrupt is generated. The configuration of FDCAN_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO
								rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus-off status
0: Bus-off status unchanged
1: Bus-off status changed

Bit 18 **EW**: Warning status
0: Error-warning status unchanged
1: Error-warning status changed

Bit 17 **EP**: Error passive
0: Error-passive status unchanged
1: Error-passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN_CCCR register.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

54.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ARAE**: Access to reserved address enable

Bit 22 **PEDE**: Protocol error in data phase enable

Bit 21 **PEAE**: Protocol error in arbitration phase enable

Bit 20 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **BOE**: Bus-off status
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **HPME**: High-priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

54.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line has to be enabled via the EINT0 and EINT1 bit of the FDCAN_ILE register.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERR:** Protocol error grouping the following interruption
ARAL: Access to reserved address line
PEDL: Protocol error in data phase line
PEAL: Protocol error in arbitration phase line
WDIL: Watchdog interrupt line
BOL: Bus-off status
EWL: Warning status interrupt line

Bit 5 **BERR:** Bit and line error grouping the following interruption
EPL Error passive interrupt line
ELOL: Error logging overflow interrupt line

Bit 4 **MISC:** Interrupt regrouping the following interruption
TOOL: Timeout occurred interrupt line
MRAFL: Message RAM access failure interrupt line
TSWL: Timestamp wraparound interrupt line

Bit 3 **TFERR:** Tx FIFO ERROR grouping the following interruption
TEFLL: Tx event FIFO element lost interrupt line
TEFFL: Tx event FIFO full interrupt line
TEFNL: Tx event FIFO new entry interrupt line
TFEL: Tx FIFO empty interrupt line

Bit 2 **SMSG:** Status message bit grouping the following interruption
TCFL: Transmission cancellation finished interrupt line
TCL: Transmission completed interrupt line
HPML: High-priority message interrupt line

Bit 1 **RXFIFO1:** RX FIFO bit grouping the following interruption
RF1LL: Rx FIFO 1 message lost interrupt line
RF1FL: Rx FIFO 1 full interrupt line
RF1NL: Rx FIFO 1 new message interrupt line

Bit 0 **RXFIFO0:** RX FIFO bit grouping the following interruption
RF0LL: Rx FIFO 0 message lost interrupt line
RF0FL: Rx FIFO 0 full interrupt line
RF0NL: Rx FIFO 0 new message interrupt line

54.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

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

Bit 1 **EINT1**: Enable interrupt line 1
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr1_it disabled
1: Interrupt line fdcan_intr1_it enabled

54.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in *Figure 787* and *Figure 788*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	LSE[3:0]				Res.	Res.	Res.	LSS[4:0]				
				rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
						rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list
0: No extended message ID filter
1 to 8: Number of extended message ID filter elements
> 8: Values greater than 8 are interpreted as 8.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list
0: No standard message ID filter
1 to 28: Number of standard message ID filter elements
> 28: Values greater than 28 are interpreted as 28.
This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **F0OM**: FIFO 0 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard
Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard IDs
1: Reject all remote frames with 11-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard IDs
1: Reject all remote frames with 29-bit standard IDs
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

54.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084

Reset value: 0x1FFF FFFF

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID mask
For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active.
This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

54.4.21 FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element:
0: Standard filter list
1: Extended filter list

Bits 14:13 Reserved, must be kept at reset value.

Bits 12:8 **FIDX[4:0]**: Filter index
Index of matching filter element.
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

54.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x0090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]	
						r	r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	F0FL[3:0]			
						r	r					r	r	r	r

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

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of the RF0L interrupt flag of the FDCAN_IR register. When RF0L is cleared, this bit is also cleared.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer.
Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]**: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0.
Range: 0 to 3.

54.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
													rw	rw	rw

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

Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index

After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN_RXF0S).

54.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]	
						r	r							r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	F1FL[3:0]			
						r	r					r	r	r	r

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

Bit 25 **RF1L**: Rx FIFO 1 message lost

This bit is a copy of the RF1L interrupt flag of the FDCAN_IR register. When RF1L is cleared, this bit is also cleared.

0: No Rx FIFO 1 message lost

1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 24 **F1F**: Rx FIFO 1 full

0: Rx FIFO 1 not full

1: Rx FIFO 1 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index

Rx FIFO 1 write index pointer.

Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F1GI[1:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer.
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1.
Range: 0 to 3.

54.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
													rw	rw	rw

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

Bits 2:0 **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN_RXF1S).

54.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

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

54.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]	
										r				r	r

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

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

Bit 21 **TFQF**: Tx FIFO/queue full
0: Tx FIFO/queue not full
1: Tx FIFO/queue full

Bits 20:18 Reserved, must be kept at reset value.

Bits 17:16 **TFQPI[1:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 3

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TFGI[1:0]**: Tx FIFO get index
Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC)

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TFFL[2:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC).

54.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8

Reset value: 0x0000 0000

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

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

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

Bits 2:0 **TRP[2:0]**: Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN_TXBCR register.

After the FDCAN_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN_TXBCF in the following cases:

- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN_TXBRP bit is cleared.

54.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC

Reset value: 0x0000 0000

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

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

Bits 2:0 **AR[2:0]**: Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

54.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0

Reset value: 0x0000 0000

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

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

Bits 2:0 **CR[2:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact.

This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. The bits remain set until the corresponding FDCAN_TXBRP bit is cleared.

0: No cancellation pending

1: Cancellation pending

54.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4

Reset value: 0x0000 0000

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

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

Bits 2:0 **TO[2:0]**: Transmission occurred.

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

54.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8

Reset value: 0x0000 0000

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

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

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

Bits 2:0 **CF[2:0]**: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN_TXBAR register.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

54.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC

Reset value: 0x0000 0000

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

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

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

Bits 2:0 **TIE[2:0]**: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

54.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

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

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

Bits 2:0 **CFIE[2:0]**: Cancellation finished interrupt enable.

Each Tx buffer has its own CFIE bit.

0: Cancellation finished interrupt disabled

1: Cancellation finished interrupt enabled

54.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4

Reset value: 0x0000 0000

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

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

Bit 25 **TEFL**: Tx event FIFO element lost

This bit is a copy of the TEFL interrupt flag of the FDCAN_IR. When TEFL is cleared, this bit is also cleared.

0 No Tx event FIFO element lost

1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full

0: Tx event FIFO not full

1: Tx event FIFO full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **EFPI[1:0]**: Event FIFO put index

Tx event FIFO write index pointer.

Range: 0 to 3.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **EFGI[1:0]**: Event FIFO get index
Tx event FIFO read index pointer.
Range: 0 to 3.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **EFFL[2:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO.
Range: 0 to 3.

54.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00E8

Reset value: 0x0000 0000

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

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

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

Bits 1:0 **EFAI[1:0]**: Event FIFO acknowledge index
After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) to EFAI[1:0] + 1 and updates the FIFO 0 fill level (EFFL[2:0] of FDCAN_TXEFS).

54.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

Address offset: 0x0100

Reset value: 0x0000 0000

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

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

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

Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE bit of the FDCAN_CCCR register is set.

Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN_CKDIV register, which changes clock divider for all instances.

If several FDCAN instances are present, the input clock divider must be modified before configuring the other FDCAN instances.

54.4.38 FDCAN register map

Table 607. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	0	0	0	0	1	1	1	1	0	1	0	1	0	0	1	1	0	1	1	1	1	1	0	1	0	0	0	1	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
	Reset value									0	0	0	0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				

Table 607. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]						
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003C	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRSRESI1	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0		0	0	0	0	0	0	0	0
0x004C	Reserved	Reserved																															
0x0050	FDCAN_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	TOO	MRAF	TSW	TEFL	TEFF	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1N	RF0L	RF0F	RF0N
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1NE	RF0LE	RF0FE	RF0NE
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 607. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0058	FDCAN_ILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERR	BERR	MISC	TFERR	SMSG	RXFIFO1	RXFIFO0
	Reset value	0																									0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x0060-0x007C	Reserved	Reserved																															
0x0080	FDCAN_RXGFC	Res.	Res.	Res.	Res.	LSE[3:0].				Res.	Res.	Res.	LSS[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	F0OM	F1OM	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
	Reset value					0	0	0	0				0	0	0	0	0							0	0			0	0	0	0	0	0
0x0084	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0088	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	Res.	Res.	FIDX[4:0]						MSI[1:0]		Res.	Res.	Res.	BIDX[2:0]		
	Reset value																	0			0	0	0	0	0	0	0	0				0	0	0
0x0090	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	Res.	Res.	Res.	Res.	F0PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F0GI[1:0]		Res.	Res.	Res.	Res.	Res.	F0FL[3:0]		
	Reset value							0	0							0	0							0	0						0	0	0	0
0x0094	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[2:0]		
	Reset value																														0	0	0
0x0098	FDCAN_RXF1S	Res.	Res.	Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	Res.	Res.	Res.	Res.	F1PI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	F1GI[1:0]		Res.	Res.	Res.	Res.	Res.	F1FL[3:0]		
	Reset value							0	0							0	0							0	0						0	0	0	0
0x009C	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[2:0]		
	Reset value																														0	0	0
0x00A0-0x00BC	Reserved	Reserved																															
0x00C0	FDCAN_TXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0	0																							
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	Res.	Res.	Res.	TFQPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	TFGI[1:0]		Res.	Res.	Res.	Res.	Res.	TFFL[2:0]		
	Reset value											0				0	0							0	0						0	1	1

Table 607. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C8	FDCAN_TXBRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRP2	TRP1	TRP0
	Reset value																														0	0	0
0x00CC	FDCAN_TXBAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AR2	AR1	AR0
	Reset value																														0	0	0
0x00D0	FDCAN_TXBCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CR2	CR1	CR0
	Reset value																														0	0	0
0x00D4	FDCAN_TXBTO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TO2	TO1	TO0
	Reset value																														0	0	0
0x00D8	FDCAN_TXBCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CF2	CF1	CF0
	Reset value																														0	0	0
0x00DC	FDCAN_TXBTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE2	TIE1	TIE0
	Reset value																														0	0	0
0x00E0	FDCAN_TXBCIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFIE2	CFIE1	CFIE0
	Reset value																														0	0	0
0x00E4	FDCAN_TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	Res.	Res.	Res.	EFPI[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	EFG[1:0]		Res.	Res.	Res.	Res.	Res.	EFFL[2:0]		
	Reset value							0	0							0	0							0	0						0	0	0
0x00E8	FDCAN_TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[1:0]	
	Reset value																															0	0
0x0100	FDCAN_CKDIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PDIV[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

55 Universal serial bus full-speed host/device interface (USB)

55.1 USB introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB2 bus.

USB suspend/resume are supported, which permits to stop the device clocks for low-power consumption.

55.2 USB main features

- USB specification version 2.0 full-speed compliant
- Supports both Host and Device modes
- Configurable number of endpoints from 1 to 8
- Dedicated packet buffer memory (SRAM) of 2048 bytes
- Cyclic redundancy check (CRC) generation/checking, Non-return-to-zero Inverted (NRZI) encoding/decoding and bit-stuffing
- Isochronous transfers support
- Double-buffered bulk/isochronous endpoint/channel support
- USB Suspend/Resume operations
- Frame locked clock pulse generation
- USB 2.0 Link Power Management support (Device mode only)
- Battery Charging Specification Revision 1.2 support (Device mode only)
- USB connect / disconnect capability (controllable embedded pull-up resistor on USB_DP line)

55.3 USB implementation

Table 608 describes the USB implementation in the devices.

Table 608. STM32H5 USB implementation

USB features(1)	USB
Host mode	X
Number of endpoints	8
Size of dedicated packet buffer memory SRAM	2048 bytes
Dedicated packet buffer memory SRAM access scheme	32 bits
USB 2.0 Link Power Management (LPM) support in device	X
Battery Charging Detection (BCD) support for device	X
Embedded pull-up resistor on USB_DP line	X

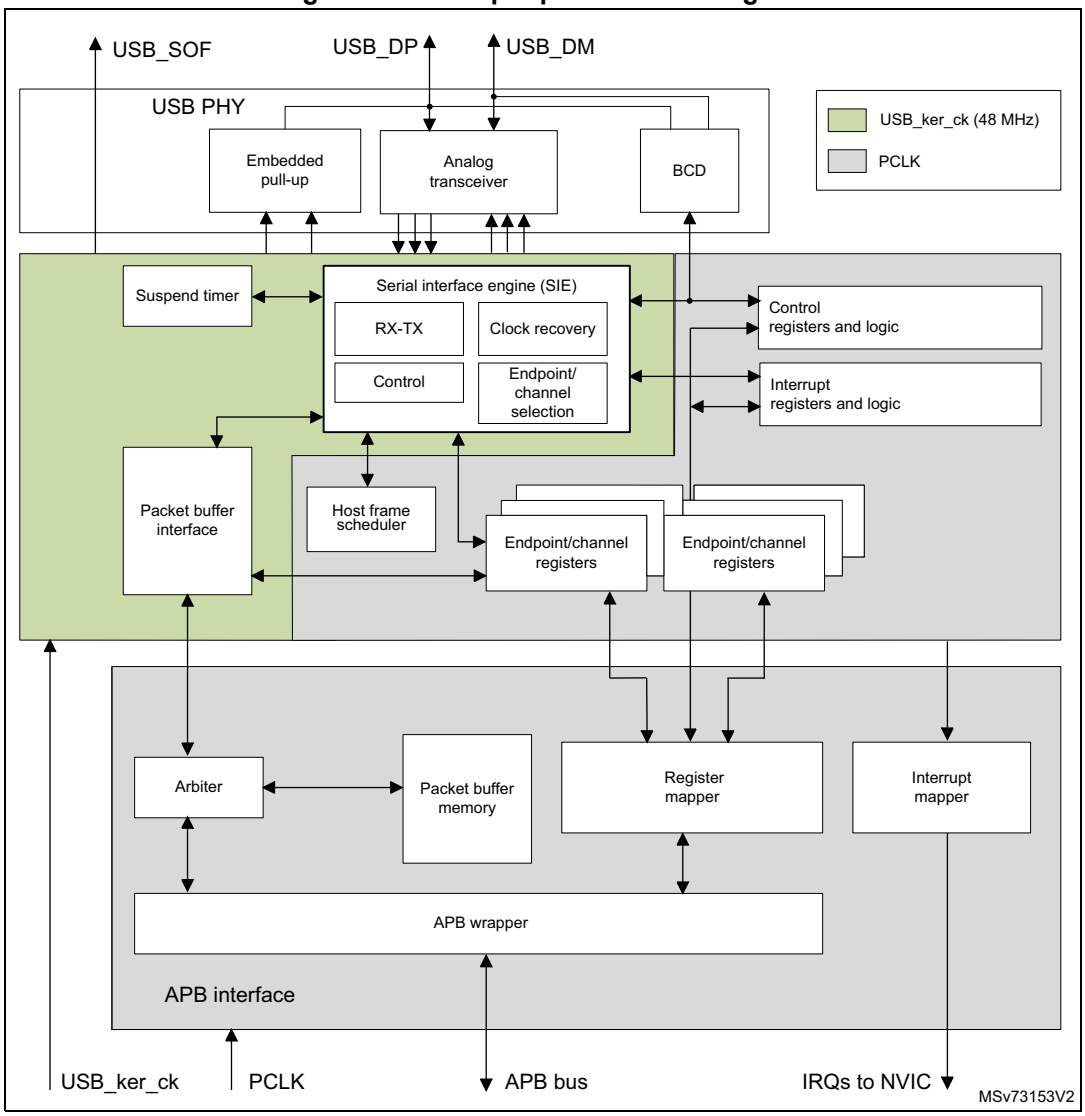
1. X= supported

55.4 USB functional description

55.4.1 USB block diagram

Figure 789 shows the block diagram of the USB peripheral.

Figure 789. USB peripheral block diagram



55.4.2 USB pins and internal signals

Table 609. USB input/output pins

Pin name	Pin type	Description
USB_DP	Digital input/output	D+ line
USB_DM	Digital input/output	D- line
USB_SOF	Digital output	SOF (start of frame indicator)

55.4.3 USB reset and clocks

A single reset is present on USB. The RCC allows a reset to be forced by software.

There are two clocks:

- PCLK for the APB bus interface and registers.
- USB_ker_ck (48 MHz) for the main protocol logic including notably the serial interface engine (SIE), see USB_ker_ck clock domain in block diagram.

55.4.4 General description and Device mode functionality

The USB peripheral provides a USB-compliant connection between the function implemented by the microcontroller and an external USB function which can be a host PC but also a USB Device. Data transfer between the external USB host or device and the system memory occurs through a dedicated packet buffer memory accessed directly by the USB peripheral. This dedicated memory size is 2048 bytes, and up to 16 mono-directional or 8 bidirectional endpoints can be used. The USB peripheral interfaces with the external USB Host or Device, detecting token packets, handling data transmission/reception, and processing handshake packets as required by the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint/channel is associated with a buffer description block indicating where the endpoint/channel-related memory area is located, how large it is or how many bytes must be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required and if the endpoint/channel is configured) takes place. The data buffered by the USB peripheral are loaded in an internal 16-bit register and memory access to the dedicated buffer is performed. When all the data have been transferred, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint/channel-specific interrupt is generated, reading status registers and/or using different interrupt response routines. The microcontroller can determine:

- which endpoint/channel has to be served,
- which type of transaction took place, if errors occurred (bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.).

Special support is offered to isochronous transfers and high throughput bulk transfers, implementing a double buffer usage, which permits to always have an available buffer for the USB peripheral while the microcontroller uses the other one.

A special bit THR512 in register USB_ISTR allows notification of 512 bytes being received in (or transmitted from) the buffer. This bit must be used for long ISO packets (from 512 to 1023 bytes) as it facilitates early start or read/write of data. In this way, the first 512 bytes can be handled by software while avoiding use of double buffer mode. This bit works when only one ISO endpoint is configured.

The unit can be placed in low-power mode (SUSPEND mode), by writing in the control register, whenever required. At this time, all static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs, while in low-power mode, wakes the device up asynchronously. A special interrupt source can be connected directly to a wake-up line to permit the system to immediately restart the normal clock generation and/or support direct clock start/stop.

Host mode and specific functionality

A single bit, HOST, in register USB_CNTR permits Host mode to be activated. Host mode functionality permits the USB to talk to a remote peripheral. Supported functionality is aligned to Device mode and uses the same register structures to manage the buffers. The same number of endpoints can be supported in Host mode, however in Host mode the terminology “channel” is preferred, as each channel is in reality a combination of the connected device and the endpoint on that device. The basic mechanisms for packet transmission and reception are the same as those supported in Device mode.

When operating in Host mode, the USB is in charge of the bus and in order to do this must issue transaction requests corresponding to active periodic and non-periodic endpoints. A host frame scheduler assures efficient use of the frame. Connection to hubs is supported. Connection to low speed devices is supported, both with a direct connection and through a hub.

Double-buffered mode, as previously described in Device mode, is also supported in Host mode, in both bulk and isochronous channels. The THR512 functionality is also supported (but as in Device mode) only for ISO traffic.

Note: Unlike in Device mode, where there is a detection of battery charging capability (in order to facilitate fast charging), there is no integrated support in Host mode to present battery charging capability (CDP or DCP cases in the standard), the host port is always presented as a default standard data port (SDP).

For LPM (link power management) this feature is not supported in Host mode.

55.4.5 Description of USB blocks used in both Device and Host modes

The USB peripheral implements all the features related to USB interfacing, which include the following blocks:

- USB physical interface (USB PHY): this block is maintaining the electrical interface to an external USB host. It contains the differential analog transceiver itself, controllable embedded pull-up resistor (connected to USB_DP line) and support for battery charging detection (BCD), multiplexed on same USB_DP and USB_DM lines.
- Serial interface engine (SIE): the functions of this block include: synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and uses the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events, such as start of frame (SOF), USB_Reset, data errors etc. and to endpoint related events like end of transmission or correct reception of a packet; these signals are then used to generate interrupts.
- Timer: this block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no traffic has been received for 3 ms.
- Packet buffer interface: this block manages the local memory implementing a set of buffers in a flexible way, both for transmission and reception. It can choose the proper buffer according to requests coming from the SIE and locate them in the memory addresses pointed by the endpoint/channel registers. It increments the address after each exchanged byte until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer to overrun the maximum capacity.

- Endpoint/channel-related registers: each endpoint/channel has an associated register containing the endpoint/channel type and its current status. For mono-directional/single-buffer endpoints, a single register can be used to implement two distinct endpoints. The number of registers is 8, allowing up to 16 mono-directional/single-buffer or up to 7 double-buffer endpoints in any combination. For example the USB peripheral can be programmed to have 4 double buffer endpoints and 8 single-buffer/mono-directional endpoints.
- Control registers: these are the registers containing information about the status of the whole USB peripheral and used to force some USB events, such as resume and power-down.
- Interrupt registers: these contain the interrupt masks and a record of the events. They can be used to inquire an interrupt reason, the interrupt status or to clear the status of a pending interrupt.

*Note: * Endpoint/channel 0 is always used for control transfer in single-buffer mode.*

The USB peripheral is connected to the APB2 bus through an APB2 interface, containing the following blocks:

- Packet memory: this is the local memory that physically contains the packet buffers. It can be used by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 2048 bytes, structured as 512 words of 32 bits.
- Arbiter: this block accepts memory requests coming from the APB2 bus and from the USB interface. It resolves the conflicts by giving priority to APB2 accesses, while always reserving half of the memory bandwidth to complete all USB transfers. This time-duplex scheme implements a virtual dual-port SRAM that allows memory access, while an USB transaction is happening. Multiword APB2 transfers of any length are also allowed by this scheme.
- Register mapper: this block collects the various byte-wide and bit-wide registers of the USB peripheral in a structured 32-bit wide word set addressed by the APB2.
- APB2 wrapper: this provides an interface to the APB2 for the memory and register. It also maps the whole USB peripheral in the APB2 address space.
- Interrupt mapper: this block is used to select how the possible USB events can generate interrupts and map them to the NVIC.

55.4.6 Description of host frame scheduler (HFS) specific to Host mode

The host frame scheduler is the hardware machine in charge to submit host channel requests on the bus according to the USB priority order and bandwidth access rules.

Host channels are divided in two categories:

- Periodic channels: isochronous and interrupt traffic types. With guaranteed bandwith access.
- Non-periodic channels: bulk and control traffic types. With best effort service.

The host frame scheduler organizes the full-speed frame in 3 sequential windows

- Periodic service window
- Non-periodic service window
- Black security window

At the start of a new frame, the host scheduler:

1. First considers all periodic channels which were active (STAT bits VALID) at the start of frame
2. Executes single round of service of periodic channels, the periodic service window, in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first
3. When the periodic round is finished, HFS closes the periodic service window and stops servicing periodic traffic even if some periodic channel was re-enabled or some new channel was enabled after the SOF.
4. Starts servicing all non-periodic channels which are currently active (STAT bits VALID) in hardware priority order from CH#1 to CH#8. For bidirectional channels it executes the OUT direction first.
5. Executes multiple round-robin service cycles of non-periodic channels until almost the end of frame
6. Non-periodic traffic can be requested at any time and is serviced by HFS with best effort latency, with the exception of a black security window at the end of the frame where new injected requests are directly postponed to the next frame to avoid babbles. This is also true for pending transactions which have not been serviced ahead of the security window.

55.5 Programming considerations for Device and Host modes

In the following sections, the expected interactions between the USB peripheral and the application program are described, in order to ease application software development.

55.5.1 Generic USB Device programming

This part describes the main tasks required of the application software in order to obtain USB compliant behavior. The actions related to the most general USB events are taken into account and paragraphs are dedicated to the special cases of double-buffered endpoints and isochronous transfers. Apart from system reset, an action is always initiated by the USB peripheral, driven by one of the USB events described below.

55.5.2 System and power-on reset

Upon system and power-on reset, the first operation the application software must perform is to provide all required clock signals to the USB peripheral and subsequently de-assert its reset signal so to be able to access its registers. The whole initialization sequence is hereafter described.

As a first step application software needs to activate register macrocell clock and de-assert macrocell specific reset signal using related control bits provided by device clock management logic.

After that, the analog part of the device related to the USB transceiver must be switched on using the PDWN bit in CNTR register, which requires a special handling. This bit is intended to switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is not defined. It is thus necessary to wait this time, after setting the PDWN bit in the CNTR register, before removing the reset condition on the USB part (by clearing the USBRST bit in the CNTR register). Clearing the ISTR register removes any spurious pending interrupt before any other macrocell operation is enabled.

At system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and data transfers. All registers not specific to any endpoint/channel must be initialized according to the needs of application software (choice of enabled interrupts, chosen address of packet buffers, etc.). Then the process continues as for the USB reset case (see further paragraph).

USB bus reset (RST_DCON interrupt) in Device mode

When this event occurs, the USB peripheral is put in the same conditions it is left by the system reset after the initialization described in the previous paragraph: communication is disabled in all endpoint registers (the USB peripheral does not respond to any packet). As a response to the USB reset event, the USB function must be enabled, having as USB address 0, implementing only the default control endpoint (endpoint address is 0 too). This is accomplished by setting the enable function (EF) bit of the USB_DADDR register and initializing the CHEP0R register and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADD[6:0] bits of the USB_DADDR register, and configures any other necessary endpoint.

When a RST_DCON interrupt is received, the application software is responsible to enable again the default endpoint of USB function 0 within 10 ms from the end of the reset sequence which triggered the interrupt.

USB bus reset in Host mode

In Host mode a bus reset is activated by setting the USBRST bit of the USB_CNTR register. It must subsequently be cleared by software once the minimum active reset time from the standard has been respected.

Structure and usage of packet buffers

Each bidirectional endpoint can receive or transmit data over the bus. The received data is stored in a dedicated memory buffer reserved for that endpoint, while another memory buffer contains the data to be transmitted by the endpoint. Access to this memory is performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgment. Since the packet buffer memory has also to be accessed by the microcontroller, an arbitration logic takes care of the access conflicts, using half APB2 cycle for microcontroller access and the remaining half for the USB peripheral access. In this way, both agents can operate as if the packet memory would be a dual-port SRAM, without being aware of any conflict even when the microcontroller is performing back-to-back accesses. The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed by the requirements of the USB standard at 48 MHz, and this can be different from the clock used for the interface to the APB2 bus. Different clock configurations are possible where the APB2 clock frequency can be higher or lower than the USB peripheral one.

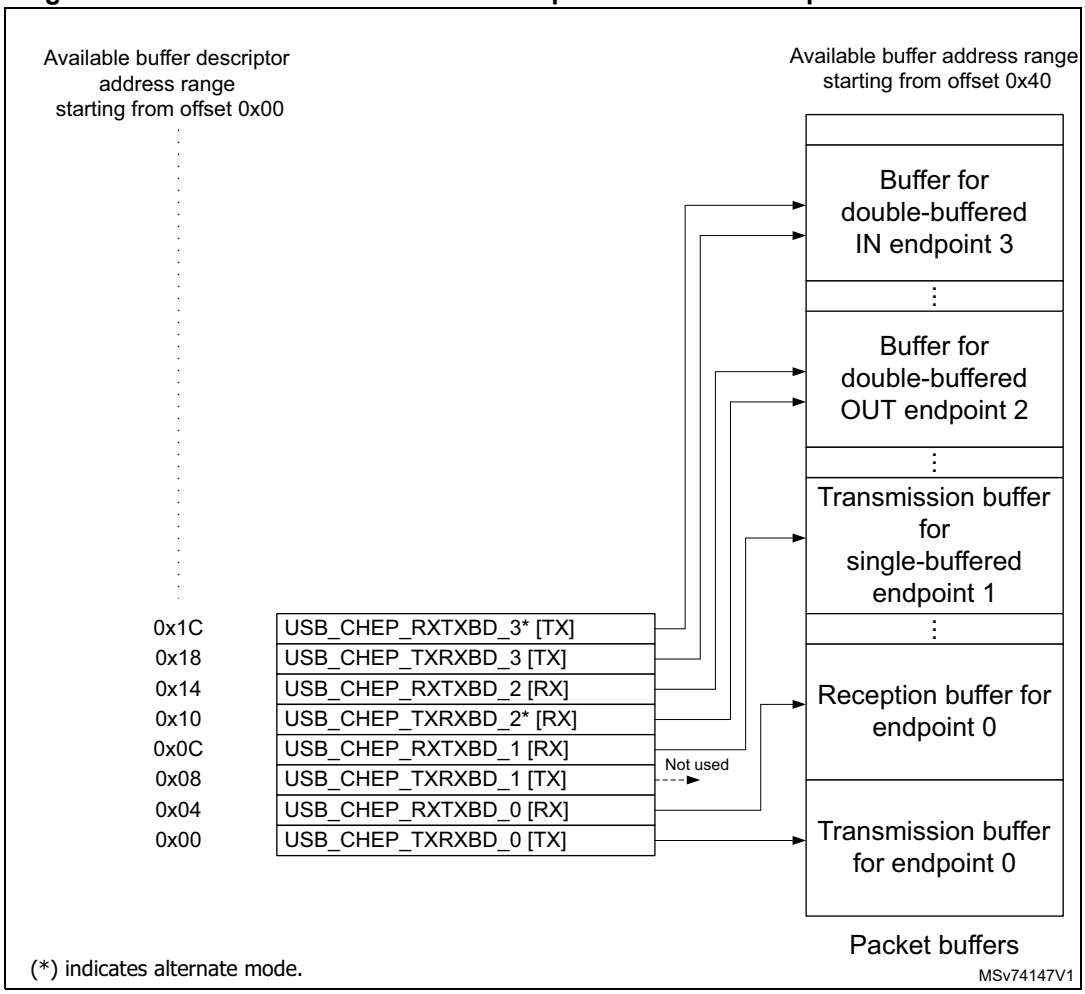
Note: Due to USB data rate and packet memory interface requirements, the APB2 clock must have a minimum frequency of 12 MHz to avoid data overrun/underrun problems.

Each endpoint is associated with two packet buffers (usually one for transmission and the other one for reception). Buffers can be placed anywhere inside the packet memory because their location and size is specified in a buffer description table, which is also located in the packet memory. Each table entry is associated to an endpoint register and it is composed of two 32-bit words so that table start address must always be aligned to an 8-

byte boundary. Buffer descriptor table entries are described in *Section 55.7: USBSRAM registers*. If an endpoint is unidirectional and it is neither an isochronous nor a double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction). Other table locations related to unsupported transfer directions or unused endpoints, are available to the user. Isochronous and double-buffered bulk endpoints have special handling of packet buffers (Refer to *Section 55.5.5: Isochronous transfers in Device mode* and *Section 55.5.3: Double-buffered endpoints and usage in Device mode* respectively). The relationship between buffer description table entries and packet buffer areas is depicted in *Figure 790*.

For Host mode different sections explain the buffer usage model, notably *Section 55.5.6: Isochronous transfers in Host mode* and *Section 55.5.4: Double buffered channels: usage in Host mode*.

Figure 790. Packet buffer areas with examples of buffer description table locations



Each packet buffer is used either during reception or transmission starting from the bottom. The USB peripheral never changes the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received (buffer overrun condition) the data is copied to the memory only up to the last available location.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADDRn_TX/ADDRn_RX fields in the CHEP_TXBD_n and CHEP_RXBD_n registers (in SRAM) so that the USB peripheral finds the data to be transmitted already available and the data to be received can be buffered. The UTYPE bits in the USB_CHEPnR register must be set according to the endpoint type, eventually using the EPKIND bit to enable any special required feature. On the transmit side, the endpoint must be enabled using the STATTX bits in the USB_CHEPnR register and COUNTn_TX must be initialized. For reception, STATRX bits must be set to enable reception and COUNTn_RX must be written with the allocated buffer size using the BLSIZE and NUM_BLOCK fields. Unidirectional endpoints, except isochronous and double-buffered bulk endpoints, need to initialize only bits and registers related to the supported direction. Once the transmission and/or reception are enabled, register USB_CHEPnR and locations ADDRn_TX/ADDRn_RX, COUNTn_TX/COUNTn_RX (respectively), must not be modified by the application software, as the hardware can change their value on the fly. When the data transfer operation is completed, notified by a CTR interrupt event, they can be accessed again to re-enable a new operation.

Data transmission in Device mode (IN packets)

When receiving an IN token packet, if the received address matches a configured and valid endpoint, the USB peripheral accesses the contents of CHEP_TXBD_n (fields ADDRn_TX and COUNTn_TX) inside the buffer descriptor table entry related to the addressed endpoint. The content of these locations is stored in its internal 16-bit registers ADDR and COUNT (not accessible by software). The packet memory is accessed again to read the first byte to be transmitted (refer to *Structure and usage of packet buffers on page 2607*) and the USB peripheral starts sending a DATA0 or DATA1 PID according to USB_CHEPnR bit DTOGTX. When the PID is completed, the first byte, read from buffer memory, is loaded into the output shift register to be transmitted on the USB bus. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to STATTX bits in the USB_CHEPnR register.

The ADDRn_TX field in the internal register CHEP_TXBD_n is used as a pointer to the current buffer memory location while COUNT is used to count the number of remaining bytes to be transmitted. Each half-word read from the packet buffer memory is transmitted over the USB bus starting from the least significant byte. Transmission buffer memory is read starting from the address pointed by ADDRn_TX for COUNTn_TX/4 words. If a transmitted packet is composed of an odd number of bytes, only the lower half of the last half-word accessed is used.

On receiving the ACK receipt by the host, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled, the endpoint is made invalid by setting STATTX = 10 (NAK) and bit VTTX is set. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. Servicing of the VTTX event starts, clearing the interrupt bit; the application software then prepares another buffer full of data to be sent, updates the COUNTn_TX table location with the number of byte to be transmitted during the next transfer, and finally sets STATTX to 11 (VALID) to re-enable transmission. While the STATTX bits are equal to 10 (NAK), any IN request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second IN transaction addressed to the same endpoint immediately following the one which triggered the CTR interrupt.

Data transmission in Host mode (OUT packets)

Data transmission in Host mode follows the same general principles as Device mode. The main differences are due to the protocol. For example the host initiates the transmission whereas the device responds to the incoming token.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents of an OUT packet are then written to that address in the packet memory and COUNTn_TX must be updated (when necessary) to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATTX must be set to 11 (VALID) in order to trigger the transmit. The transmission is then scheduled by the HFS.

After a successful transmission the CTR interrupt (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATTX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATTX is now in NAK. In the case of a STALL response, STATTX is in STALL. In this last case, the bus must be reset.

On receiving the ACK receipt by the device, the USB_CHEPnR register is updated in the following way: DTOGTX bit is toggled.

An error condition is signaled via the bits VTTX and ERR_TX if one of the following conditions occurs:

- No handshake being received in time
- False EOP
- Bit stuffing error
- Invalid handshake PID

Data reception in Device mode (OUT and SETUP packets)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following paragraph about control transfers. When receiving an OUT/SETUP PID, if the address matches a valid endpoint, the USB peripheral accesses the contents of the ADDRn_RX and COUNTn_RX fields inside the buffer descriptor table entry related to the addressed endpoint. The content of the ADDRn_RX field is stored directly in its internal register ADDR. Internal register COUNT is now reset and the values of BLSIZE and NUM_BLOCK bit fields, which are read within USB_CHEP_RXBD_n content, are used to initialize BUF_COUNT, an internal 16-bit counter, which is used to check the buffer overrun condition (all these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in half-words (the first byte received is stored as least significant byte) and then transferred to the packet buffer starting from the address contained in the internal ADDR register while BUF_COUNT is decremented and COUNT is incremented at each byte transfer. When the end of DATA packet is detected, the correctness of the received CRC is tested and only if no errors occurred during the reception, an ACK handshake packet is sent back to the transmitting host.

In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied in the packet memory buffer, at least until the error detection point, but the ACK packet is not sent and the ERR bit in USB_ISTR register is set. However, there is usually no software action required in this case: the USB peripheral recovers from reception errors and remains ready for the next transaction to come. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the ACK, according to bits

STATRX in the USB_CHEPnR register, and no data is written in the reception memory buffers.

Reception memory buffer locations are written starting from the address contained in the ADDRn_RX for a number of bytes corresponding to the received data packet length, or up to the last allocated memory location, as defined by BLSIZE and NUM_BLOCK, whichever comes first. In this way, the USB peripheral never writes beyond the end of the allocated reception memory buffer area. If the length of the data packet payload (actual number of bytes used by the application) is greater than the allocated buffer, the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent instead of the usual ACK to notify the problem to the host, no interrupt is generated and the transaction is considered failed.

When the transaction is completed correctly, by sending the ACK handshake packet, the internal COUNT register is copied back in the COUNTn_RX location inside the buffer description table entry, leaving unaffected BLSIZE and NUM_BLOCK fields, which normally do not require to be re-written, and the USB_CHEPnR register is updated in the following way: DTOGRX bit is toggled, the endpoint is made invalid by setting STATRX = 10 (NAK) and bit VTRX is set. If the transaction has failed due to errors or buffer overrun condition, none of the previously listed actions take place. The application software must first identify the endpoint, which is requesting microcontroller attention by examining the IDN and DIR bits in the USB_ISTR register. The VTRX event is serviced by first determining the transaction type (SETUP bit in the USB_CHEPnR register); the application software must clear the interrupt flag bit and get the number of received bytes reading the COUNTn_RX location inside the buffer description table entry related to the endpoint being processed. After the received data is processed, the application software must set the STATRX bits to 11 (VALID) in the USB_CHEPnR, enabling further transactions. While the STATRX bits are equal to 10 (NAK), any OUT request addressed to that endpoint is NAKed, indicating a flow control condition: the USB host retries the transaction until it succeeds. It is mandatory to execute the sequence of operations in the above mentioned order to avoid losing the notification of a second OUT transaction addressed to the same endpoint following immediately the one which triggered the CTR interrupt.

Data reception in Host mode (IN packets)

Data reception in Host mode follows the same general principles as Device mode. The main differences are again due to the protocol. In the device, data can be received or not, depending on readiness after previous operations, whereas the host only requests receive data when it is ready and able to store them.

ADDRn_TX must be set to the location in the packet memory reserved for the packet for transmission. The contents received in the data phase response to the IN token packet are then written to that address in the packet memory and COUNTn_TX gets updated by hardware during this process to indicate the number of bytes in the packet.

DEVADDR must be written for the correct endpoint and then STATRX must be set to VALID in order to trigger the reception. The reception is then scheduled by the HFS.

After a successful reception the interrupt CTR (correct transfer) is triggered. By examining IDN and DIR bits, the corresponding channel and direction is understood. On the indicated channel, the STATRX field now has transitioned to DISABLE. In the case of a NAK being received (when the peripheral is not ready) STATRX now is in NAK. In the case of a STALL response, STATRX is in STALL. In this last case, the bus must be reset. During an IN packet

an error condition is signaled via the bits VTRX and ERR_RX if one of the following conditions occurs:

- False EOP
- Bit stuffing error
- Wrong CRC

Control transfers in Device mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only and are very similar to OUT ones (data reception) except that the values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 1 and 0 respectively, to initialize the control transfer, and both STATTX and STATRX are set to 10 (NAK) to let software decide if subsequent transactions must be IN or OUT depending on the SETUP contents. A control endpoint must check SETUP bit in the USB_CHEPnR register at each VTRX event to distinguish normal OUT transactions from SETUP ones. A USB Device can determine the number and direction of data stages by interpreting the data transferred in the SETUP stage, and is required to STALL the transaction in the case of errors. To do so, at all data stages before the last, the unused direction must be set to STALL, so that, if the host reverses the transfer direction too soon, it gets a STALL as a status stage.

While enabling the last data stage, the opposite direction must be set to NAK, so that, if the host reverses the transfer direction (to perform the status stage) immediately, it is kept waiting for the completion of the control operation. If the control operation completes successfully, the software changes NAK to VALID, otherwise to STALL. At the same time, if the status stage is an OUT, the STATUS_OUT (EPKIND in the USB_CHEPnR register) bit must be set, so that an error is generated if a status transaction is performed with non-zero data. When the status transaction is serviced, the application clears the STATUS_OUT bit and sets STATRX to VALID (to accept a new command) and STATTX to NAK (to delay a possible status stage immediately following the next setup).

Since the USB specification states that a SETUP packet cannot be answered with a handshake different from ACK, eventually aborting a previously issued command to start the new one, the USB logic does not permit a control endpoint to answer with a NAK or STALL packet to a SETUP token received from the host.

When the STATRX bits are set to 01 (STALL) or 10 (NAK) and a SETUP token is received, the USB accepts the data, performing the required data transfers and sends back an ACK handshake. If that endpoint has a previously issued VTRX request not yet acknowledged by the application (for example VTRX bit is still set from a previously completed reception), the USB discards the SETUP transaction and does not answer with any handshake packet regardless of its state, simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint immediately following the transaction, which triggered the VTRX interrupt.

Control transfers in Host mode

Control transfers are made of a SETUP transaction, followed by zero or more data stages, all of the same direction, followed by a status stage (a zero-byte transfer in the opposite direction). SETUP transactions are handled by control endpoints only. A control endpoint must set the SETUP bit in the USB_CHEPnR register. The values of DTOGTX and DTOGRX bits of the addressed endpoint registers are set to 0. Depending on whether it is a

control write or control read then STATTX or STATRX are set to 11 (ACTIVE) in order to trigger the control transfer via the host frame scheduler.

On receiving a CTR interrupt the channel (device address and endpoint) can be determined by examining IDN and DIR bits. Devices are expected to NAK every control unless the packet is corrupted in which case they do not acknowledge. The situation is reflected in the value of STATTX.

In the case of an error condition the ERR bit gets set. One possible case is where a CRC error is seen at the device, in this case no ACK is returned to the host. The host sees no ACK and after an appropriate delay this generates a timeout error with ERR_TX set (which can generate an interrupt).

55.5.3 Double-buffered endpoints and usage in Device mode

All different endpoint types defined by the USB standard represent different traffic models, and describe the typical requirements of different kind of data transfer operations. When large portions of data are to be transferred between the host PC and the USB function, the bulk endpoint type is the most suited model. This is because the host schedules bulk transactions so as to fill all the available bandwidth in the frame, maximizing the actual transfer rate as long as the USB function is ready to handle a bulk transaction addressed to it. If the USB function is still busy with the previous transaction when the next one arrives, it answers with a NAK handshake and the host PC issues the same transaction again until the USB function is ready to handle it, reducing the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called ‘double-buffering’ can be used with bulk endpoints.

When ‘double-buffering’ is activated, data toggle sequencing is used to select, which buffer is to be used by the USB peripheral to perform the required data transfers, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to always have a complete buffer to be used by the application, while the USB peripheral fills the other one. For example, during an OUT transaction directed to a ‘reception’ double-buffered bulk endpoint, while one buffer is being filled with new data coming from the USB host, the other one is available for the microcontroller software usage (the same would happen with a ‘transmission’ double-buffered bulk endpoint and an IN transaction).

Since the swapped buffer management requires the usage of all 4 buffer description table locations hosting the address pointer and the length of the allocated memory buffers, the USB_CHEPnR registers used to implement double-buffered bulk endpoints are forced to be used as unidirectional ones. Therefore, only one STAT bit pair must be set at a value different from 00 (DISABLED): STATRX if the double-buffered bulk endpoint is enabled for reception, STATTX if the double-buffered bulk endpoint is enabled for transmission. In case it is required to have double-buffered bulk endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

To exploit the double-buffering feature and reach the highest possible transfer rate, the endpoint flow control structure, described in previous chapters, has to be modified, in order to switch the endpoint status to NAK only when a buffer conflict occurs between the USB peripheral and application software, instead of doing it at the end of each successful transaction. The memory buffer which is currently being used by the USB peripheral is defined by the DTOG bit related to the endpoint direction: DTOGRX (bit 14 of USB_CHEPnR register) for ‘reception’ double-buffered bulk endpoints or DTOGTX (bit 6 of USB_CHEPnR register) for ‘transmission’ double-buffered bulk endpoints. To implement the new flow control scheme, the USB peripheral must know which packet buffer is currently in

use by the application software, so to be aware of any conflict. Since in the USB_CHEPnR register, there are two DTOG bits but only one is used by USB peripheral for data and buffer sequencing (due to the unidirectional constraint required by double-buffering feature) the other one can be used by the application software to show which buffer it is currently using. This new buffer flag is called SW_BUF. In the following table the correspondence between USB_CHEPnR register bits and DTOG/SW_BUF definition is explained, for the cases of 'transmission' and 'reception' double-buffered bulk endpoints.

Table 610. Double-buffering buffer flag definition

Buffer flag	'Transmission' endpoint	'Reception' endpoint
DTOG	DTOGTX (USB_CHEPnR bit 6)	DTOGRX (USB_CHEPnR bit 14)
SW_BUF	USB_CHEPnR bit 14	USB_CHEPnR bit 6

The memory buffer which is currently being used by the USB peripheral is defined by DTOG buffer flag, while the buffer currently in use by application software is identified by SW_BUF buffer flag. The relationship between the buffer flag value and the used packet buffer is the same in both cases, and it is listed in the following table.

Table 611. Bulk double-buffering memory buffers usage (Device mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (IN)	0	1	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	1	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

Double-buffering feature for a bulk endpoint is activated by performing the two following actions:

- Writing UTYPE bit field at 00 in its USB_CHEPnR register, to define the endpoint as a bulk
- Setting EPKIND bit at 1 (DBL_BUF), in the same register.

The application software is responsible for DTOG and SW_BUF bits initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. The end of the first transaction occurring after having set DBL_BUF, triggers the special flow control of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF remain set. At the end of each transaction the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making the USB peripheral buffer swapping completely software independent. Unlike common transactions, and the first one after DBL_BUF setting, STAT bit pair is not affected by the transaction termination and its value remains 11 (VALID). However, as the token packet of a new transaction is received, the actual endpoint status is masked as 10 (NAK) when a buffer conflict between the USB peripheral and the application software is detected (this condition is identified by DTOG and SW_BUF having the same value, see *Table 611*). The application software responds to the CTR event notification by clearing the interrupt flag and starting any required handling of the completed transaction. When the application packet buffer usage is over, the software toggles the SW_BUF bit, writing 1 to it, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is limited only by the application elaboration time of a transaction data: if the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions due to flow control takes place and the actual transfer rate is limited only by the host PC.

The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from 11 (VALID) into the STAT bit pair of the related USB_CHEPnR register. In this case, the USB peripheral always uses the programmed endpoint status, regardless of the buffer usage condition.

55.5.4 Double buffered channels: usage in Host mode

In Host mode the underlying transmit and receive methods for double buffered channels are the same as those described for Device mode.

Similar to the Device mode table, a new table below *Table 612: Bulk double-buffering memory buffers usage (Host mode)* shows the programming settings for OUT and IN tokens.

Table 612. Bulk double-buffering memory buffers usage (Host mode)

Endpoint type	DTOG	SW_BUF	Packet buffer used by USB peripheral	Packet buffer used by Application Software
Transmit (OUT)	0	1	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	0	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (IN)	0	1	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	0	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	0	0	None (1)	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	1	None (1)	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

1. Endpoint in NAK Status.

55.5.5 Isochronous transfers in Device mode

The USB standard supports full speed peripherals requiring a fixed and accurate data production/consume frequency, defining this kind of traffic as ‘isochronous’. Typical examples of this data are: audio samples, compressed video streams, and in general any sort of sampled data having strict requirements for the accuracy of delivered frequency. When an endpoint is defined to be ‘isochronous’ during the enumeration phase, the host allocates in the frame the required bandwidth and delivers exactly one IN or OUT packet each frame, depending on endpoint direction. To limit the bandwidth requirements, no re-transmission of failed transactions is possible for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase and no ACK packet is expected or sent after the data packet. For the same reason, isochronous transfers do not support data toggle sequencing and always use DATA0 PID to start any data packet.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID), any other value produces results not compliant to USB standard. Isochronous endpoints implement double-buffering

to ease application software development, using both ‘transmission’ and ‘reception’ packet memory areas to manage buffer swapping on each successful transaction in order to have always a complete buffer to be used by the application, while the USB peripheral fills the other.

The memory buffer which is currently used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOGRX for ‘reception’ isochronous endpoints, DTOGTX for ‘transmission’ isochronous endpoints, both in the related USB_CHEPnR register) according to *Table 613*.

Table 613. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (IN)	0	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations
	1	USB_CHEP_RXTXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDR_TX / COUNT_TX) Buffer description table locations.
Receive (OUT)	0	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDR_RX / COUNT_RX) Buffer description table locations.

As it happens with double-buffered bulk endpoints, the USB_CHEPnR registers used to implement isochronous endpoints are forced to be used as unidirectional ones. In case it is required to have isochronous endpoints enabled both for reception and transmission, two USB_CHEPnR registers must be used.

The application software is responsible for the DTOG bit initialization according to the first buffer to be used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the VTRX or VTTX bit of the addressed endpoint USB_CHEPnR register is set, depending on the enabled direction. At the same time, the affected DTOG bit in the USB_CHEPnR register is hardware toggled making buffer swapping completely software independent. STAT bit pair is not affected by transaction completion; since no flow control is possible for isochronous transfers due to the lack of handshake phase, the endpoint remains always 11 (VALID). CRC errors or buffer-overrun conditions occurring during isochronous OUT transfers are anyway considered as correct transactions and they always trigger a VTRX event. However, CRC errors set the ERR bit in the USB_ISTR register anyway, in order to notify the software of the possible data corruption.

55.5.6 Isochronous transfers in Host mode

From the host point of view isochronous packets are issued or requested one by frame by the host frame scheduler. There is no NAK/ACK protocol and no resend of data or token.

The mechanism is based on a table very similar to that for Device mode. See *Table 614* to understand the relationship between the DTOG bit buffers and the buffer usage.

Table 614. Isochronous memory buffers usage

Endpoint Type	DTOG bit value	Packet buffer used by the USB peripheral	Packet buffer used by the application software
Transmit (OUT)	0	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.	USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations
	1	USB_CHEP_RXTXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations	USB_CHEP_TXRXBD_0 (ADDRn_TX / COUNTn_TX) Buffer description table locations.
Receive (IN)	0	USB_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.
	1	USB_CHEP_TXRXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations	USB_CHEP_RXTXBD_0 (ADDRn_RX / COUNTn_RX) Buffer description table locations.

The isochronous behavior for an endpoint is selected by setting the UTYPE bits at 10 in its USB_CHEPnR register; since there is no handshake phase the only legal values for the STATRX/STATTX bit pairs are 00 (DISABLED) and 11 (VALID),

Just as in Device mode, the mechanism allows automatic toggle of the DTOG bit. Note that in Host mode, at the same time as this toggle, the STATTX or STATRX of the completed buffer is automatically set to DISABLED, permitting the future buffer to be accessed before re-enabling it by setting it to 11 (VALID).

55.5.7 Suspend/resume events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus must not be greater than 2.5 mA. This requirement is of fundamental importance for bus-powered devices, while self-powered devices are not required to comply to this strict power consumption constraint. In suspend mode, the host PC sends the notification by not sending any traffic on the USB bus for more than 3 ms: since a SOF packet must be sent every 1 ms during normal operations, the USB peripheral detects the lack of 3 consecutive SOF packets as a suspend request from the host PC and set the SUSP bit to 1 in USB_ISTR register, causing an interrupt if enabled. Once the device is suspended, its normal operation can be restored by a so called RESUME sequence, which can be started from the host PC or directly from the peripheral itself, but it is always terminated by the host PC. The suspended USB peripheral must be anyway able to detect a RESET sequence, reacting to this event as a normal USB reset event.

The actual procedure used to suspend the USB peripheral is device dependent since according to the device composition, different actions may be required to reduce the total consumption.

A brief description of a typical suspend procedure is provided below, focused on the USB-related aspects of the application software routine responding to the SUSP notification of the USB peripheral:

1. Set the SUSPEN bit in the USB_CNTR register to 1. This action activates the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSP interrupts being issued while the USB is suspended.
2. Remove or reduce any static power consumption in blocks different from the USB peripheral.
3. Set SUSPRDY bit in USB_CNTR register to 1 to remove static power consumption in the analog USB transceivers but keeping them able to detect resume activity.
4. Optionally turn off external oscillator and device PLL to stop any activity inside the device.

When an USB event occurs while the device is in SUSPEND mode, the RESUME procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular care must be taken to ensure that this process does not take more than 10 ms when the wakening event is an USB reset sequence (see “Universal Serial Bus Specification” for more details). The start of a resume or reset sequence, while the USB peripheral is suspended, clears the SUSPRDY bit in USB_CNTR register asynchronously. Even if this event can trigger a WKUP interrupt if enabled, the use of an interrupt response routine must be carefully evaluated because of the long latency due to system clock restart; to have the shorter latency before re-activating the nominal clock it is suggested to put the resume procedure just after the end of the suspend one, so its code is immediately executed as soon as the system clock restarts. To prevent ESD discharges or any other kind of noise from waking-up the system (the exit from suspend mode is an asynchronous event), a suitable analog filter on data line status is activated during suspend; the filter width is about 70 ns.

The following is a list of actions a resume procedure must address:

1. Optionally turn on external oscillator and/or device PLL.
2. Clear SUSPEN bit of USB_CNTR register.
3. If the resume triggering event has to be identified, bits RXDP and RXDM in the USB_FNR register can be used according to *Table 615*, which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected monitoring the status of the above mentioned bits by checking when they reach the “10” configuration, which represent the idle bus state; moreover at the end of a reset sequence the RST_DCON bit in USB_ISTR register is set to 1, issuing an interrupt if enabled, which must be handled as usual.

Table 615. Resume event detection

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Root reset	None
“10”	None (noise on bus)	Go back in Suspend mode
“01”	Root resume	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

A device may require to exit from suspend mode as an answer to particular events not directly related to the USB protocol (for example a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the L2RES bit in the USB_CNTR register to 1 and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed using ESOF interrupts, occurring with a 1 ms period when the system clock is running at nominal frequency). Once the L2RES bit is clear, the resume sequence is completed by the host PC and its end can be monitored again using the RXDP and RXDM bits in the USB_FNR register.

Note: The L2RES bit must be anyway used only after the USB peripheral has been put in suspend mode, setting the SUSPEN bit in USB_CNTR register to 1.

Suspend and resume in Host mode

The basics of the suspend and resume mechanism has been described in the previous section.

From the host stand-point, suspend is entered by writing the SUSPEN bit in USB_CNTR. When suspend entry is confirmed, SUSPRDY (also in USB_CNTR) is set.

Once in suspend, and when the application want to resume the bus, this can be done by setting the L2RES bit in USB_CNTR to 1.

Below in *Table 616*, the different actions recommended after a wake-up event are indicated. According to the different line states after a wake-up event, the interpretation of the event and the suggested behavior are shown. Note that, this table here is somewhat expanded when compared to the previously shown device table, as the host may encounter both full speed and low speed devices which use different line states for both suspend and resume.

Table 616. Resume event detection for host

[RXDP,RXDM] status	Wake-up event	Required resume software action
“00”	Not allowed (noise on bus)	Go back in Suspend mode
“10”	Full speed capable device: Not allowed (noise on bus) Low speed device: Device remote wake-up resume	None
“01”	Full speed capable device: Device remote wake-up resume Low speed device: Not allowed (noise on bus)	None
“11”	Not allowed (noise on bus)	Go back in Suspend mode

55.6 USB registers

The USB peripheral registers can be divided into the following groups:

- Common registers: interrupt and control registers. These registers affect the general behavior of the USB peripheral defining operating mode, interrupt handling, device address and giving access to the current frame number updated by the host PC.
  - USB_CNTR
  - USB_ISTR
  - USB_FNR
  - USB_DADDR
  - USB_LPMCSR
  - USB_BCDR
- Endpoint/channel registers: endpoint/channel configuration and status
  - USB_CHEPnR

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

55.6.1 USB control register (USB_CNTR)

Address offset: 0x40

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M
rw														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	r	rw	rw

Bit 31 **HOST:** HOST mode

HOST bit selects betweens host or device USB mode of operation. It must be set before enabling the USB peripheral by the function enable bit.

0: USB Device function

1: USB host function

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **DDISCM**: Device disconnection mask

– Host mode

0: Device disconnection interrupt disabled

1: Device disconnection interrupt enabled

Bit 16 **THR512M**: 512 byte threshold interrupt mask

0: 512 byte threshold interrupt disabled

1: 512 byte threshold interrupt enabled

Bit 15 **CTRM:** Correct transfer interrupt mask
0: Correct transfer (CTR) interrupt disabled.
1: CTR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 14 **PMAOVRM:** Packet memory area over / underrun interrupt mask
0: PMAOVR interrupt disabled.
1: PMAOVR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 13 **ERRM:** Error interrupt mask
0: ERR interrupt disabled.
1: ERR interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 12 **WKUPM:** Wake-up interrupt mask
0: WKUP interrupt disabled.
1: WKUP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 11 **SUSPM:** Suspend mode interrupt mask
0: Suspend mode request (SUSP) interrupt disabled.
1: SUSP interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 10 **RST_DCONM:** USB reset request (Device mode) or device connect/disconnect (Host mode) interrupt mask
0: RESET interrupt disabled.
1: RESET interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 9 **SOFM:** Start of frame interrupt mask
0: SOF interrupt disabled.
1: SOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 8 **ESOFM:** Expected start of frame interrupt mask
0: Expected start of frame (ESOF) interrupt disabled.
1: ESOF interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 7 **L1REQM:** LPM L1 state request interrupt mask
0: LPM L1 state request (L1REQ) interrupt disabled.
1: L1REQ interrupt enabled, an interrupt request is generated when the corresponding bit in the USB_ISTR register is set.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **L1RES:** L1 remote wake-up / resume driver
– Device mode
Software sets this bit to send a LPM L1 50 µs remote wake-up signaling to the host. After the signaling ends, this bit is cleared by hardware.
0: No effect
1: Send 50 µs remote-wake-up signaling to host

Bit 4 **L2RES:** L2 remote wake-up / resume driver

– Device mode

The microcontroller can set this bit to send remote wake-up signaling to the host. It must be activated, according to USB specifications, for no less than 1 ms and no more than 15 ms after which the host PC is ready to drive the resume sequence up to its end.

– Host mode

Software sets this bit to send resume signaling to the device.

Software clears this bit to send end of resume to device and restart SOF generation.

In the context of remote wake up, this bit is to be set following the WAKEUP interrupt.

0: No effect

1: Send L2 resume signaling to device

Bit 3 **SUSPEN:** Suspend state enable

– Condition: Device mode

Software can set this bit when the SUSP interrupt is received, which is issued when no traffic is received by the USB peripheral for 3 ms. Software can also set this bit when the L1REQ interrupt is received with positive acknowledge sent.

As soon as the suspend state is propagated internally all device activity is stopped, USB clock is gated, USB transceiver is set into low power mode and the SUSPRDY bit is set by hardware. In the case that device application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the microcontroller to stop mode, as in the case of bus powered device application, it must first wait few cycles to see the SUSPRDY = 1 acknowledge the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

– Condition: Host mode

Software can set this bit when host application has nothing scheduled for the next frames and wants to enter long term power saving. When set, it stops immediately SOF generation and any other host activity, gates the USB clock and sets the transceiver in low power mode. If any USB transaction is on-going at the time SUSPEN is set, suspend is entered at the end of the current transaction.

As soon as suspend state is propagated internally and gets effective the SUSPRDY bit is set. In the case that host application wants to pursue more aggressive power saving by stopping the USB clock source and by moving the micro-controller to STOP mode, it must first wait few cycles to see SUSPRDY=1 acknowledge to the suspend request.

This bit is cleared by hardware simultaneous with the WAKEUP flag set.

0: No effect

1: Enter L1/L2 suspend

Bit 2 **SUSPRDY:** Suspend state effective

This bit is set by hardware as soon as the suspend state entered through the SUSPEN control gets internally effective. In this state USB activity is suspended, USB clock is gated, transceiver is set in low power mode by disabling the differential receiver. Only asynchronous wake-up logic and single ended receiver is kept alive to detect remote wake-up or resume events.

Software must poll this bit to confirm it to be set before any STOP mode entry.

This bit is cleared by hardware simultaneously to the WAKEUP flag being set.

0: Normal operation

1: Suspend state

Bit 1 **PDWN:** Power down

This bit is used to completely switch off all USB-related analog parts if it is required to completely disable the USB peripheral for any reason. When this bit is set, the USB peripheral is disconnected from the transceivers and it cannot be used.
0: Exit power down
1: Enter power down mode

Bit 0 **USBRST:** USB Reset

– Condition: Device mode

Software can set this bit to reset the USB core, exactly as it happens when receiving a RESET signaling on the USB.The USB peripheral, in response to a RESET, resets its internal protocol state machine. Reception and transmission are disabled until the RST_DCON bit is cleared. All configuration registers do not reset: the microcontroller must explicitly clear these registers (this is to ensure that the RST_DCON interrupt can be safely delivered, and any transaction immediately followed by a RESET can be completed). The function address and endpoint registers are reset by an USB reset event.
0: No effect
1: USB core is under reset

– Condition: Host mode

Software sets this bit to drive USB reset state on the bus and initialize the device. USB reset terminates as soon as this bit is cleared by software.
0: No effect
1: USB reset driven

55.6.2 USB interrupt status register (USB_ISTR)

Address offset: 0x44

Reset value: 0x0000 0000

This register contains the status of all the interrupt sources permitting application software to determine which events caused an interrupt request.

The upper part of this register contains single bits, each of them representing a specific event. These bits are set by the hardware when the related event occurs; if the corresponding bit in the USB_CNTR register is set, a generic interrupt request is generated. The interrupt routine, examining each bit, performs all necessary actions, and finally it clears the serviced bits. If any of them is not cleared, the interrupt is considered to be still pending, and the interrupt line is kept high again. If several bits are set simultaneously, only a single interrupt is generated.

Endpoint/channel transaction completion can be handled in a different way to reduce interrupt response latency. The CTR bit is set by the hardware as soon as an endpoint/channel successfully completes a transaction, generating a generic interrupt request if the corresponding bit in USB_CNTR is set. An endpoint/channel dedicated interrupt condition is activated independently from the CTRM bit in the USB_CNTR register. Both interrupt conditions remain active until software clears the pending bit in the corresponding USB_CHEPnR register (the CTR bit is actually a read only bit). For endpoint-/channel-related interrupts, the software can use the direction of transaction (DIR) and IDN read-only bits to identify which endpoint/channel made the last interrupt request and called the corresponding interrupt service routine.

The user can choose the relative priority of simultaneously pending USB_ISTR events by specifying the order in which software checks USB_ISTR bits in an interrupt service routine. Only the bits related to events, which are serviced, are cleared. At the end of the service routine, another interrupt is requested, to service the remaining conditions.

To avoid spurious clearing of some bits, it is recommended to clear them with a load instruction where all bits which must not be altered are written with 1, and all bits to be cleared are written with 0 (these bits can only be cleared by software). Read-modify-write cycles must be avoided because between the read and the write operations another bit can be set by the hardware and the next write clears it before the device has the time to service the event.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LS_ DCON	DCON_ STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR 512
	r	r												rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR	PMA OVR	ERR	WKUP	SUSP	RST_ DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0			r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LS_DCON**: Low speed device connected

– Host mode:

This bit is set by hardware when an LS device connection is detected. Device connection is signaled after LS J-state is sampled for 22 consecutive cycles of the USB clock (48 MHz) from the unconnected state.

Bit 29 **DCON_STAT**: Device connection status

– Host mode:

This bit contains information about device connection status. It is set by hardware when a LS/FS device is attached to the host while it is reset when the device is disconnected.

0: No device connected

1: FS or LS device connected to the host

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **DDISC**: Device connection

– Host mode

This bit is set when a device connection is detected. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 16 **THR512**: 512 byte threshold interrupt

This bit is set to 1 by the hardware when 512 bytes have been transmitted or received during isochronous transfers. This bit is read/write but only 0 can be written and writing 1 has no effect. Note that no information is available to indicate the associated channel/endpoint, however in practice only one ISO endpoint/channel with such large packets can be supported, so that channel.

Bit 15 **CTR**: Completed transfer in host mode

This bit is set by the hardware to indicate that an endpoint/channel has successfully completed a transaction; using DIR and IDN bits software can determine which endpoint/channel requested the interrupt. This bit is read-only.

Bit 14 **PMAOVR:** Packet memory area over / underrun

This bit is set if the microcontroller has not been able to respond in time to an USB memory request. The USB peripheral handles this event in the following way: During reception an ACK handshake packet is not sent, during transmission a bit-stuff error is forced on the transmitted stream; in both cases the host retries the transaction. The PMAOVR interrupt must never occur during normal operations. Since the failed transaction is retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however this does not happen during isochronous transfers (no isochronous transaction is anyway retried) leading to a loss of data in this case. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 13 **ERR:** Error

This flag is set whenever one of the errors listed below has occurred:

NANS: No ANSwer. The timeout for a host response has expired.

CRC: Cyclic redundancy check error. One of the received CRCs, either in the token or in the data, was wrong.

BST: Bit stuffing error. A bit stuffing error was detected anywhere in the PID, data, and/or CRC.

FVIO: Framing format violation. A non-standard frame was received (EOP not in the right place, wrong token sequence, etc.).

The USB software can usually ignore errors, since the USB peripheral and the PC host manage retransmission in case of errors in a fully transparent way. This interrupt can be useful during the software development phase, or to monitor the quality of transmission over the USB bus, to flag possible problems to the user (for example loose connector, too noisy environment, broken conductor in the USB cable and so on). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 12 **WKUP:** Wake-up

This bit is set to 1 by the hardware when, during suspend mode, activity is detected that wakes up the USB peripheral. This event asynchronously clears the SUSPRDY bit in the CTLR register and activates the USB_WAKEUP line, which can be used to notify the rest of the device (for example wake-up unit) about the start of the resume process. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 11 **SUSP:** Suspend mode request

– Device mode

This bit is set by the hardware when no traffic has been received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset and it is disabled by the hardware when the suspend mode is active (SUSPEN=1) until the end of resume sequence. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 10 **RST_DCON:** USB reset request (Device mode) or device connect/disconnect (Host mode)

– Device mode

This bit is set by hardware when an USB reset is released by the host and the bus returns to idle. USB reset state is internally detected after the sampling of 60 consecutive SE0 cycles.

– Host mode

This bit is set by hardware when device connection or device disconnection is detected. Device connection is signaled after J state is sampled for 22 cycles consecutively from unconnected state. Device disconnection is signaled after SE0 state is seen for 22 bit times consecutively from connected state.

Bit 9 **SOF:** Start of frame

This bit signals the beginning of a new USB frame and it is set when a SOF packet arrives through the USB bus. The interrupt service routine can monitor the SOF events to have a 1 ms synchronization event to the USB host and to safely read the USB_FNR register which is updated at the SOF packet reception (this can be useful for isochronous applications). This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 8 **ESOF:** Expected start of frame

– Device mode

This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each 1 ms, but if the device does not receive it properly, the suspend timer issues this interrupt. If three consecutive ESOF interrupts are generated (for example three SOF packets are lost) without any traffic occurring in between, a SUSP interrupt is generated. This bit is set even when the missing SOF packets occur while the suspend timer is not yet locked. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bit 7 **L1REQ:** LPM L1 state request

– Device mode

This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged. This bit is read/write but only 0 can be written and writing 1 has no effect.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **DIR:** Direction of transaction

This bit is written by the hardware according to the direction of the successful transaction, which generated the interrupt request.
If DIR bit = 0, VTTX bit is set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of IN type (data transmitted by the USB peripheral to the host PC).
If DIR bit = 1, VTRX bit or both VTTX/VTRX are set in the USB_CHEPnR register related to the interrupting endpoint. The interrupting transaction is of OUT type (data received by the USB peripheral from the host PC) or two pending transactions are waiting to be processed. This information can be used by the application software to access the USB_CHEPnR bits related to the triggering transaction since it represents the direction having the interrupt pending. This bit is read-only.

Bits 3:0 **IDN[3:0]:** Device Endpoint / host channel identification number

These bits are written by the hardware according to the host channel or device endpoint number, which generated the interrupt request. If several endpoint/channel transactions are pending, the hardware writes the identification number related to the endpoint/channel having the highest priority defined in the following way: two levels are defined, in order of priority: isochronous and double-buffered bulk channels/endpoints are considered first and then the others are examined. If more than one endpoint/channel from the same set is requesting an interrupt, the IDN bits in USB_ISTR register are assigned according to the lowest requesting register, CHEP0R having the highest priority followed by CHEP1R and so on. The application software can assign a register to each endpoint/channel according to this priority scheme, so as to order the concurring endpoint/channel requests in a suitable way. These bits are read only.

55.6.3 USB frame number register (USB_FNR)

Address offset: 0x48

Reset value: 0x0000 0XXX (where X is undefined)

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RXDP:** Receive data + line status

This bit can be used to observe the status of received data plus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 14 **RXDM:** Receive data - line status

This bit can be used to observe the status of received data minus upstream port data line. It can be used during end-of-suspend routines to help determining the wake-up event.

Bit 13 **LCK:** Locked

– Device mode

This bit is set by the hardware when at least two consecutive SOF packets have been received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer remains in this state until an USB reset or USB suspend event occurs.

Bits 12:11 **LSOF[1:0]:** Lost SOF

– Device mode

These bits are written by the hardware when an ESOF interrupt is generated, counting the number of consecutive SOF packets lost. At the reception of an SOF packet, these bits are cleared.

Bits 10:0 **FN[10:0]:** Frame number

This bit field contains the 11-bits frame number contained in the last received SOF packet. The frame number is incremented for every frame sent by the host and it is useful for isochronous transfers. This bit field is updated on the generation of an SOF interrupt.

55.6.4 USB Device address (USB_DADDR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

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

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

Bit 7 **EF:** Enable function

This bit is set by the software to enable the USB Device. The address of this device is contained in the following ADD[6:0] bits. If this bit is at 0 no transactions are handled, irrespective of the settings of USB_CHEPnR registers.

Bits 6:0 **ADD[6:0]:** Device address

– Device mode

These bits contain the USB function address assigned by the host PC during the enumeration process. Both this field and the endpoint/channel address (EA) field in the associated USB_CHEPnR register must match with the information contained in a USB token in order to handle a transaction to the required endpoint.

– Host mode

These bits contain the address transmitted with the LPM transaction

55.6.5 USB LPM control and status register (USB_LPMCSR)

Address offset: 0x54

Reset value: 0x0000 0000

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

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

Bits 7:4 **BESL[3:0]:** BESL value

– Device mode

These bits contain the BESL value received with last ACKed LPM Token

Bit 3 **REMWAKE:** bRemoteWake value

– Device mode

This bit contains the bRemoteWake value received with last ACKed LPM Token

Bit 2 Reserved, must be kept at reset value.

Bit 1 **LPMACK:** LPM token acknowledge enable

– Device mode:

0: the valid LPM token is NYET.
1: the valid LPM token is ACK.
The NYET/ACK is returned only on a successful LPM transaction:
No errors in both the EXT token and the LPM token (else ERROR)
A valid bLinkState = 0001B (L1) is received (else STALL)

Bit 0 **LPMEN:** LPM support enable

– Device mode

This bit is set by the software to enable the LPM support within the USB Device. If this bit is at 0 no LPM transactions are handled.

55.6.6 USB battery charging detector (USB_BCDR)

Address offset: 0x58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
rw								r	r	r	r	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **DPPU_DPD:** DP pull-up / DPDM pull-down

– Device mode

This bit is set by software to enable the embedded pull-up on DP line. Clearing it to 0 can be used to signal disconnect to the host when needed by the user software.

– Host mode

This bit is set by software to enable the embedded pull-down on DP and DM lines.

Bits 14:8 Reserved, must be kept at reset value.

Bit 7 **PS2DET:** DM pull-up detection status

– Device mode

This bit is active only during PD and gives the result of comparison between DM voltage level and V_{LGC} threshold. In normal situation, the DM level must be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.

0: Normal port detected (connected to SDP, ACA, CDP or DCP).

1: PS2 port or proprietary charger detected.

Bit 6 **SDET:** Secondary detection (SD) status

– Device mode

This bit gives the result of SD.

0: CDP detected.

1: DCP detected.

Bit 5 **PDET:** Primary detection (PD) status

– Device mode

This bit gives the result of PD.

0: no BCD support detected (connected to SDP or proprietary device).

1: BCD support detected (connected to ACA, CDP or DCP).

Bit 4 **DCDET:** Data contact detection (DCD) status

– Device mode

This bit gives the result of DCD.

0: data lines contact not detected.

1: data lines contact detected.

Bit 3 **SDEN:** Secondary detection (SD) mode enable
– Device mode
This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 2 **PDEN:** Primary detection (PD) mode enable
– Device mode
This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 1 **DCDEN:** Data contact detection (DCD) mode enable
– Device mode
This bit is set by the software to put the BCD into DCD mode. Only one detection mode (DCD, PD, SD or OFF) must be selected to work correctly.

Bit 0 **BCDEN:** Battery charging detector (BCD) enable
– Device mode
This bit is set by the software to enable the BCD support within the USB Device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD must be placed in OFF mode by clearing this bit to 0 in order to allow the normal USB operation.

55.6.7 USB endpoint/channel n register (USB_CHEPnR)

Address offset: 0x00 + 0x4 * n, (n = 0 to 7)

Reset value: 0x0000 0000

The USB peripheral supports up to 8 bidirectional endpoints or host channels. Each USB Device must support a control endpoint/channel whose address (EA bits) must be set to 0. The USB peripheral behaves in an undefined way if multiple endpoints are enabled having the same endpoint/channel number value. For each endpoint, an USB_CHEPnR register is available to store the endpoint/channel specific information.

They are also reset when an USB reset is received from the USB bus or forced through bit USBRST in the CTLR register, except the VTRX and VTTX bits, which are kept unchanged to avoid missing a correct packet notification immediately followed by an USB reset event. Each endpoint/channel has its USB_CHEPnR register where *n* is the endpoint/channel identifier.

Read-modify-write cycles on these registers must be avoided because between the read and the write operations some bits can be set by the hardware and the next write would modify them before the CPU has the time to detect the change. For this purpose, all bits affected by this problem have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended to modify these registers with a load instruction where all the bits, which can be modified only by the hardware, are written with their ‘invariant’ value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	THREE_ERR_RX[1:0]		THREE_ERR_TX[1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]						
	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rw	rc_w0	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VTRX	DTOG RX	STATRX[1:0]		SETUP	UTYPE[1:0]		EP KIND	VTTX	DTOG TX	STATTX[1:0]		EA[3:0]			
rc_w0	t	t	t	r	rw	rw	rw	rc_w0	t	t	t	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **THREE_ERR_RX[1:0]:** Three errors for an IN transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an IN transaction. THREE_ERR_RX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bits 28:27 **THREE_ERR_TX[1:0]:** Three errors for an OUT or SETUP transaction

– Host mode

This bit is set by the hardware when 3 consecutive transaction errors occurred on the USB bus for an OUT transaction. THREE_ERR_TX is not generated for isochronous transactions. The software can only clear this bit.

Coding of the received error:

00: Less than 3 errors received.

01: More than 3 errors received, last error is timeout error.

10: More than 3 errors received, last error is data error (CRC error).

11: More than 3 errors received, last error is protocol error (invalid PID, false EOP, bitstuffing error, SYNC error).

Bit 26 **ERR_RX:** Received error for an IN transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an IN transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 25 **ERR_TX:** Received error for an OUT/SETUP transaction

– Host mode

This bit is set by the hardware when an error (for example no answer by the device, CRC error, bit stuffing error, framing format violation, etc.) has occurred during an OUT or SETUP transaction on this channel. The software can only clear this bit. If the ERRM bit in USB_CNTR register is set, a generic interrupt condition is generated together with the channel related flag, which is always activated.

Bit 24 **LS_EP:** Low speed endpoint – host with HUB only

– Host mode

This bit is set by the software to send an LS transaction to the corresponding endpoint.
0: Full speed endpoint
1: Low speed endpoint

Bit 23 **NAK:**

– Host mode

This bit is set by the hardware when a device responds with a NAK. Software can use this bit to monitor the number of NAKs received from a device.

Bits 22:16 **DEVADDR[6:0]:**

– Host mode

Device address assigned to the endpoint during the enumeration process.

Bit 15 **VTRX:** USB valid transaction received

– Device mode

This bit is set by the hardware when an OUT/SETUP transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated. The type of occurred transaction, OUT or SETUP, can be determined from the SETUP bit described below.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written, writing 1 has no effect.

– Host mode

This bit is set by the hardware when an IN transaction is successfully completed on this channel. The software can only clear this bit. If the CTRM bit in USB_CNTR register is set a generic interrupt condition is generated together with the channel related flag, which is always activated.

- A transaction ended with a NAK sets this bit and NAK answer is reported to application reading the NAK state from the STATRX field of this register. One NAKed transaction keeps pending and is automatically retried by the host at the next frame, or the host can immediately retry by resetting STATRX state to VALID.

- A transaction ended by STALL handshake sets this bit and the STALL answer is reported to application reading the STALL state from the STATRX field of this register. Host application must consequently disable the channel and re-enumerate.

- A transaction ended with ACK handshake sets this bit
If double buffering is disabled, ACK answer is reported by application reading the DISABLE state from the STATRX field of this register. Host application must read received data from USBRAM and re-arm the channel by writing VALID to the STATRX field of this register.
If double buffering is enabled, ACK answer is reported by application reading VALID state from the STATRX field of this register. Host application must read received data from USBRAM and toggle the DTOGTX bit of this register.

- A transaction ended with error sets this bit.
Errors can be seen via the bits ERR_RX (host mode only).

This bit is read/write but only 0 can be written, writing 1 has no effect.

Bit 14 **DTOGRX:** Data Toggle, for reception transfers

If the endpoint/channel is not isochronous, this bit contains the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. Hardware toggles this bit, when the ACK handshake is sent following a data packet reception having a matching data PID value; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP PID received from host (in device mode), while it sets this bit to 1 when SETUP transaction is acknowledged by device (in host mode).
If the endpoint/channel is using the double-buffering feature this bit is used to support packet buffer swapping too (Refer to *Section 55.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used only to support packet buffer swapping for data transmission since no data toggling is used for this kind of channels/endpoints and only DATA0 packet are transmitted (Refer to *Section 55.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet reception, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint is not a control one) or to force specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGRX remains unchanged, while writing 1 makes the bit value toggle. This bit is read/write but it can be only toggled by writing 1.

Bits 13:12 **STATRX[1:0]:** Status bits, for reception transfers

– Device mode

These bits contain information about the endpoint status, which are listed in *Table 617: Reception status encoding on page 2639*. These bits can be toggled by software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATRX bits to NAK when a correct transfer has occurred (VTRX = 1) corresponding to a OUT or SETUP (control only) transaction addressed to this endpoint, so the software has the time to elaborate the received data before it acknowledges a new transaction.

Double-buffered bulk endpoints implement a special transaction flow control, which control the status based upon buffer availability condition (Refer to *Section 55.5.3: Double-buffered endpoints and usage in Device mode*).

If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so that the hardware cannot change the status of the endpoint after a successful transaction. If the software sets the STATRX bits to ‘STALL’ or ‘NAK’ for an isochronous endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

These bits are the host application controls to start, retry, or abort host transactions driven by the channel.

These bits also contain information about the device answer to the last IN channel transaction and report the current status of the channel according to the following STATRX table of states:

- DISABLE

  DISABLE value is reported in case of ACK acknowledge is received on a single-buffer channel. When in DISABLE state the channel is unused or not active waiting for application to restart it by writing VALID. Application can reset a VALID channel to DISABLE to abort a transaction. In this case the transaction is immediately removed from the host execution list. If the aborted transaction was already under execution it is regularly terminated on the USB but the relative VTRX interrupt is not generated.

- VALID

  A host channel is actively trying to submit USB transaction to device only when in VALID state.VALID state can be set by software or automatically by hardware on a NAKED channel at the start of a new frame. When set to VALID, an host channel enters the host execution queue and waits permission from the host frame scheduler to submit its configured transaction.

  VALID value is also reported in case of ACK acknowledge is received on a double-buffered channel. In this case the channel remains active on the alternate buffer while application needs to read the current buffer and toggle DTOGTX. In case software is late in reading and the alternate buffer is not ready, the host channel is automatically suspended transparently to the application. The suspended double buffered channel is re-activated as soon as delay is recovered and DTOGTX is toggled.

- NAK

  NAK value is reported in case of NAK acknowledge received. When in NAK state the channel is suspended and does not try to transmit. NAK state is moved to VALID by hardware at the start of the next frame, or software can change it to immediately retry transmission by writing it to VALID, or can disable it and abort the transaction by writing DISABLE

- STALL

  STALL value is reported in case of STALL acknowledge received. When in STALL state the channel behaves as disabled. Application must not retry transmission but reset the USB and re-enumerate.

Bit 11 **SETUP:** Setup transaction completed

– Device mode

This bit is read-only and it is set by the hardware when the last completed transaction is a SETUP. This bit changes its value only for control endpoints. It must be examined, in the case of a successful receive transaction (VTRX event), to determine the type of transaction occurred. To protect the interrupt service routine from the changes in SETUP bits due to next incoming tokens, this bit is kept frozen while VTRX bit is at 1; its state changes when VTRX is at 0. This bit is read-only.

– Host mode

This bit is set by the software to send a SETUP transaction on a control endpoint. This bit changes its value only for control endpoints. It is cleared by hardware when the SETUP transaction is acknowledged and VTTX interrupt generated.

Bits 10:9 **UTYPE[1:0]:** USB type of transaction

These bits configure the behavior of this endpoint/channel as described in *Table 618: Endpoint/channel type encoding*. Channel0/Endpoint0 must always be a control endpoint/channel and each USB function must have at least one control endpoint/channel which has address 0, but there can be other control channels/endpoints if required. Only control channels/endpoints handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint/channel is defined as NAK, the USB peripheral does not answer, simulating a receive error, in the receive direction when a SETUP transaction is received. If the control endpoint/channel is defined as STALL in the receive direction, then the SETUP packet is accepted anyway, transferring data and issuing the CTR interrupt. The reception of OUT transactions is handled in the normal way, even if the endpoint/channel is a control one. Bulk and interrupt endpoints have very similar behavior and they differ only in the special feature available using the EPKIND configuration bit.
The usage of isochronous channels/endpoints is explained in *Section 55.5.5: Isochronous transfers in Device mode*

Bit 8 **EPKIND:** endpoint/channel kind

The meaning of this bit depends on the endpoint/channel type configured by the UTYPE bits. *Table 619* summarizes the different meanings.
DBL_BUF: This bit is set by the software to enable the double-buffering feature for this bulk endpoint. The usage of double-buffered bulk endpoints is explained in *Section 55.5.3: Double-buffered endpoints and usage in Device mode*.
STATUS_OUT: This bit is set by the software to indicate that a status out transaction is expected: in this case all OUT transactions containing more than zero data bytes are answered ‘STALL’ instead of ‘ACK’. This bit can be used to improve the robustness of the application to protocol errors during control transfers and its usage is intended for control endpoints only. When STATUS_OUT is reset, OUT transactions can have any number of bytes, as required.

Bit 7 **VTTX:** Valid USB transaction transmitted

– Device mode

This bit is set by the hardware when an IN transaction is successfully completed on this endpoint; the software can only clear this bit. If the CTRM bit in the USB_CNTR register is set accordingly, a generic interrupt condition is generated together with the endpoint related interrupt condition, which is always activated.
A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred, as in the case of protocol errors or data toggle mismatches.
This bit is read/write but only 0 can be written.

– Host mode

Same as VTRX behavior but for USB OUT and SETUP transactions.

Bit 6 **DTOGTX:** Data toggle, for transmission transfers

If the endpoint/channel is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. Hardware toggles this bit when the ACK handshake is received from the USB host, following a data packet transmission. If the endpoint/channel is defined as a control one, hardware sets this bit to 1 at the reception of a SETUP PID addressed to this endpoint (in device mode) or when a SETUP transaction is acknowledged by the device (in host mode).
If the endpoint/channel is using the double buffer feature, this bit is used to support packet buffer swapping too (Refer to *Section 55.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint/channel is isochronous, this bit is used to support packet buffer swapping since no data toggling is used for this sort of endpoints and only DATA0 packet are transmitted (refer to *Section 55.5.5: Isochronous transfers in Device mode*). Hardware toggles this bit just after the end of data packet transmission, since no handshake is used for isochronous transfers.
This bit can also be toggled by the software to initialize its value (mandatory when the endpoint/channel is not a control one) or to force a specific data toggle/packet buffer usage. When the application software writes 0, the value of DTOGTX remains unchanged, while writing 1 makes the bit value to toggle. This bit is read/write but it can only be toggled by writing 1.

Bits 5:4 **STATTX[1:0]:** Status bits, for transmission transfers

– Device mode

These bits contain the information about the endpoint status, listed in *Table 620*. These bits can be toggled by the software to initialize their value. When the application software writes 0, the value remains unchanged, while writing 1 makes the bit value to toggle. Hardware sets the STATTX bits to NAK, when a correct transfer has occurred (VTTX = 1) corresponding to a IN or SETUP (control only) transaction addressed to this channel/endpoint. It then waits for the software to prepare the next set of data to be transmitted.
Double-buffered bulk endpoints implement a special transaction flow control, which controls the status based on buffer availability condition (Refer to *Section 55.5.3: Double-buffered endpoints and usage in Device mode*).
If the endpoint is defined as isochronous, its status can only be “VALID” or “DISABLED”. Therefore, the hardware cannot change the status of the channel/endpoint/channel after a successful transaction. If the software sets the STATTX bits to ‘STALL’ or ‘NAK’ for an isochronous channel/endpoint, the USB peripheral behavior is not defined. These bits are read/write but they can be only toggled by writing 1.

– Host mode

The STATTX bits contain the information about the channel status. Refer to *Table 620* for the full descriptions (“Host mode” descriptions). Whereas in Device mode, these bits contain the status that are given out on the following transaction, in Host mode they capture the status last received from the device. If a NAK is received, STATTX contains the value indicating NAK.

Bits 3:0 **EA[3:0]:** endpoint/channel address

– Device mode

Software must write in this field the 4-bit address used to identify the transactions directed to this endpoint. A value must be written before enabling the corresponding endpoint.

– Host mode

Software must write in this field the 4-bit address used to identify the channel addressed by the host transaction.

Table 617. Reception status encoding

STATRX[1:0]	Meaning
00	**DISABLED:** all reception requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all reception requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.
10	**NAK**: Device mode: the endpoint is NAKed and all reception requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the reception request.
11	**VALID**: this endpoint/channel is enabled for reception.

Table 618. Endpoint/channel type encoding

UTYPE[1:0]	Meaning
00	BULK
01	CONTROL
10	ISO
11	INTERRUPT

Table 619. Endpoint/channel kind meaning

UTYPE[1:0]		EPKIND meaning
00	BULK	DBL_BUF
01	CONTROL	STATUS_OUT
10	ISO	SBUF_ISO: This bit is set by the software to enable the single-buffering feature for isochronous endpoint
11	INTERRUPT	Not used

Table 620. Transmission status encoding

STATTX[1:0]	Meaning
00	**DISABLED:** all transmission requests addressed to this endpoint/channel are ignored.
01	**STALL**: Device mode: the endpoint is stalled and all transmission requests result in a STALL handshake. Host mode: this indicates that the device has STALLed the channel.

Table 620. Transmission status encoding (continued)

STATTX[1:0]	Meaning
10	**NAK**: Device mode: the endpoint is NAKed and all transmission requests result in a NAK handshake. Host mode: this indicates that the device has NAKed the transmission request.
11	**VALID**: this endpoint/channel is enabled for transmission.

55.6.8 USB register map

The table below provides the USB register map and reset values.

Table 621. USB register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USB_CHEP0R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	**USB_CHEP1R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	**USB_CHEP2R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	**USB_CHEP3R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	**USB_CHEP4R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	**USB_CHEP5R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	**USB_CHEP6R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	**USB_CHEP7R**	Res.	THREE_ERR_RX [1:0]		THREE_ERR_TX [1:0]		ERR_RX	ERR_TX	LS_EP	NAK	DEVADDR[6:0]							VTRX	DTOGRX	STATRX [1:0]		SETUP	UTYPE [1:0]		EPKIND	VTTX	DTOGTX	STATTX [1:0]		EA[3:0]			
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 621. USB register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x20-0x3F	Reserved																																
0x40	**USB_CNTR**	HOST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISCM	THR512M	CTRM	PMAOVRM	ERRM	WKUPM	SUSPM	RST_DCONM	SOFM	ESOFM	L1REQM	Res.	L1RES	L2RES	SUSPEN	SUSPRDY	PDWN	USBRST
	Reset value	0														0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	1	1
0x44	**USB_ISTR**	Res.	LS_DCON	DCON_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DDISC	THR512	CTR	PMAOVR	ERR	WKUP	SUSP	RST_DCON	SOF	ESOF	L1REQ	Res.	Res.	DIR	IDN[3:0]			
	Reset value		0	0												0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x48	**USB_FNR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDP	RXDM	LCK	LSOF[1:0]		FN[10:0]										
	Reset value																	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x
0x4C	**USB_DADDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EF	ADD[6:0]						
	Reset value																									0	0	0	0	0	0	0	0
0x54	**USB_LPMCSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BESL[3:0]				REMWAKE	Res.	LPMACK	LPMEN
	Reset value																									0	0	0	0	0		0	0
0x58	**USB_BCDR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DPPU_DPD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PS2DET	SDET	PDET	DCDET	SDEN	PDEN	DCDEN	BCDEN
	Reset value																	0								0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

55.7 USBSRAM registers

Note: The buffer descriptor table is located inside the packet buffer memory in the separate "USBSRAM" address space.

Although the buffer descriptor table is located inside the packet buffer memory ("USBSRAM" area), its entries can be considered as additional registers used to configure the location and size of the packet buffers used to exchange data between the USB macro cell and the device.

The first packet memory location is located at USBSRAM base address. The buffer descriptor table entry associated with the USB_CHEPnR registers is described below. The memory must be addressed using Word (32-bit) accesses.

A thorough explanation of packet buffers and the buffer descriptor table usage can be found in *Structure and usage of packet buffers on page 2607*.

55.7.1 Channel/endpoint transmit buffer descriptor n (USB_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program the use of double buffering working in receive mode (otherwise refer to following register description)

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

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

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

55.7.2 Channel/endpoint receive buffer descriptor n [alternate] (USB_CHEP_TXRXBD_n)

Address offset: 0x0 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs the use of double buffering and activates receive buffers (otherwise refer to previous register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLSIZE	NUM_BLOCK[4:0]					COUNT_RX[9:0]									
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

– If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.

– If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 622*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

Table 622. Definition of allocated buffer memory

Value of NUM_BLOCK[4:0]	Memory allocated when BLSIZE=0	Memory allocated when BLSIZE=1
0 (00000)	Not allowed	32 bytes
1 (00001)	2 bytes	64 bytes
2 (00010)	4 bytes	96 bytes
3 (00011)	6 bytes	128 bytes
...	...	...
14 (01110)	28 bytes	480 bytes
15 (01111)	30 bytes	
16 (10000)	32 bytes	
...	...	...
29 (11101)	58 bytes	...
30 (11110)	60 bytes	992 bytes
31 (11111)	62 bytes	1023 bytes

55.7.3 Channel/endpoint receive buffer descriptor n (USB_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register does not program use of double buffering in the transmit mode (otherwise refer to following register description).

This table location is used to store two different values, both required during packet reception. The most significant bits contains the definition of allocated buffer size, to allow buffer overflow detection, while the least significant part of this location is written back by the USB peripheral at the end of reception to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented using the number of allocated memory blocks, where block size can be selected to choose the trade-off between fine-granularity/small-buffer and coarse-granularity/large-buffer. The size of allocated buffer is a part of the endpoint/channel descriptor and it is normally defined during the enumeration process according to its maxPacketSize parameter value (see “Universal Serial Bus Specification”).

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

Bit 31 **BLSIZE:** Block size

This bit selects the size of memory block used to define the allocated buffer area.

- If BLSIZE = 0, the memory block is 2-byte large, which is the minimum block allowed in a half-word wide memory. With this block size the allocated buffer size ranges from 2 to 62 bytes.
- If BLSIZE = 1, the memory block is 32-byte large, which permits to reach the maximum packet length defined by USB specifications. With this block size the allocated buffer size theoretically ranges from 32 to 1024 bytes, which is the longest packet size allowed by USB standard specifications. However, the applicable size is limited by the available buffer memory.

Bits 30:26 **NUM_BLOCK[4:0]:** Number of blocks

These bits define the number of memory blocks allocated to this packet buffer. The actual amount of allocated memory depends on the BLSIZE value as illustrated in *Table 622*.

Bits 25:16 **COUNT_RX[9:0]:** Reception byte count

These bits contain the number of bytes received by the endpoint/channel associated with the USB_CHEPnR register during the last OUT/SETUP transaction addressed to it.

Note: Although the application only needs to read this value, it is writable.

Bits 15:0 **ADDR_RX[15:0]:** Reception buffer address

These bits point to the starting address of the packet buffer, which contains the data received by the endpoint/channel associated with the USB_CHEPnR register at the next OUT/SETUP token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

55.7.4 Channel/endpoint transmit buffer descriptor n [alternate] (USB_CHEP_RXTXBD_n)

Address offset: 0x4 + 0x8 * n, (n = 0 to 7)

Reset value: 0xXXXX XXXX

This register description applies when corresponding CHEPnR register programs use of double buffering and activates transmit buffers (otherwise refer to previous register description).

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

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

Bits 25:16 **COUNT_TX[9:0]:** Transmission byte count

These bits contain the number of bytes to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it.

Bits 15:0 **ADDR_TX[15:0]:** Transmission buffer address

These bits point to the starting address of the packet buffer containing data to be transmitted by the endpoint/channel associated with the USB_CHEPnR register at the next IN token addressed to it. Bits 1 and 0 must always be written as “00” since packet memory is word wide and all packet buffers must be word aligned.

55.7.5 USBSRAM register map

The table below provides the USB register map and reset values.

Table 623. USBSRAM register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	**USB_ CHEP_ TXRXBD_0**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x00	**USB_ CHEP_ TXRXBD_0 [alternate]**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	**USB_ CHEP_ RXTXBD_0**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04	**USB_ CHEP_ RXTXBD_0 [alternate]**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	**USB_ CHEP_ TXRXBD_1**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	**USB_ CHEP_ TXRXBD_1 [alternate]**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	**USB_ CHEP_ RXTXBD_1**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0C	**USB_ CHEP_ RXTXBD_1 [alternate]**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
. . . .	. . . .	. . . .																															
0x38	**USB_ CHEP_ TXRXBD_7**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x38	**USB_ CHEP_ TXRXBD_7 [alternate]**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	**USB_ CHEP_ RXTXBD_7**	BLSIZE	NUM_ BLOCK[4:0]					COUNT_RX[9:0]										ADDR_RX[15:0]															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x3C	**USB_ CHEP_ RXTXBD_7 [alternate]**	Res.	Res.	Res.	Res.	Res.	Res.	COUNT_TX[9:0]										ADDR_TX[15:0]															
	Reset value							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

56 USB Type-C®/USB Power Delivery interface (UCPD)

56.1 UCPD introduction

The USB Type-C/USB Power Delivery interface complies with:

- Universal Serial Bus Type-C Cable and Connector Specification: release 2.4, Oct 2024
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.2, version 1.1, October 2024

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

56.2 UCPD main features

- Compliance with USB Type-C specification release 2.4
- Compliance with USB Power Delivery specifications revision 2.0 and 3.2
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - “Dead battery” Rd support
  - USB Power Delivery message transmission and reception
  - FRS (fast role swap) Rx support
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS signaling
  - FRS detection, generating an interrupt
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

56.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.

In the following table, “X” denotes *supported*, and “-” *not supported*.

Table 624. UCPD implementation

UCPD feature	UCPD1
Dead battery support via UCPDx_DBCC1 and UCPDx_DBCC2 external signals	X
UCPDx_FRSTX as alternate function pin	X
Fully automatic trimming	-(1)
USB PD receiver hardware filter control	X
Discrete-component PHY support	-

1. Apply software trimming as described in *Section 56.5.5: UCPD software trimming*.

The following table gives the memory locations of trim data stored in the non-volatile memory, to use in the software trimming procedure described in *Section 56.5.5: UCPD software trimming*.

Table 625. UCPD software trim data

Name	Non-volatile memory location	
	Address	Bits
3A0_CC1[3:0]	0x4002 242C	7:4
3A0_CC2[3:0]	0x4002 242C	15:12
1A5_CC1[3:0]	0x08FF F844	3:0
1A5_CC2[3:0]	0x08FF F844	19:16
Rd_CC1[3:0]	0x4002 242C	3:0
Rd_CC2[3:0]	0x4002 242C	11:8

56.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), BIST test data (Tx and Rx), and Power Delivery Rx FRS signaling.

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD_FRSTX pin (alternate function), external NMOS transistors that ensure low-resistance pull-down on CC lines.

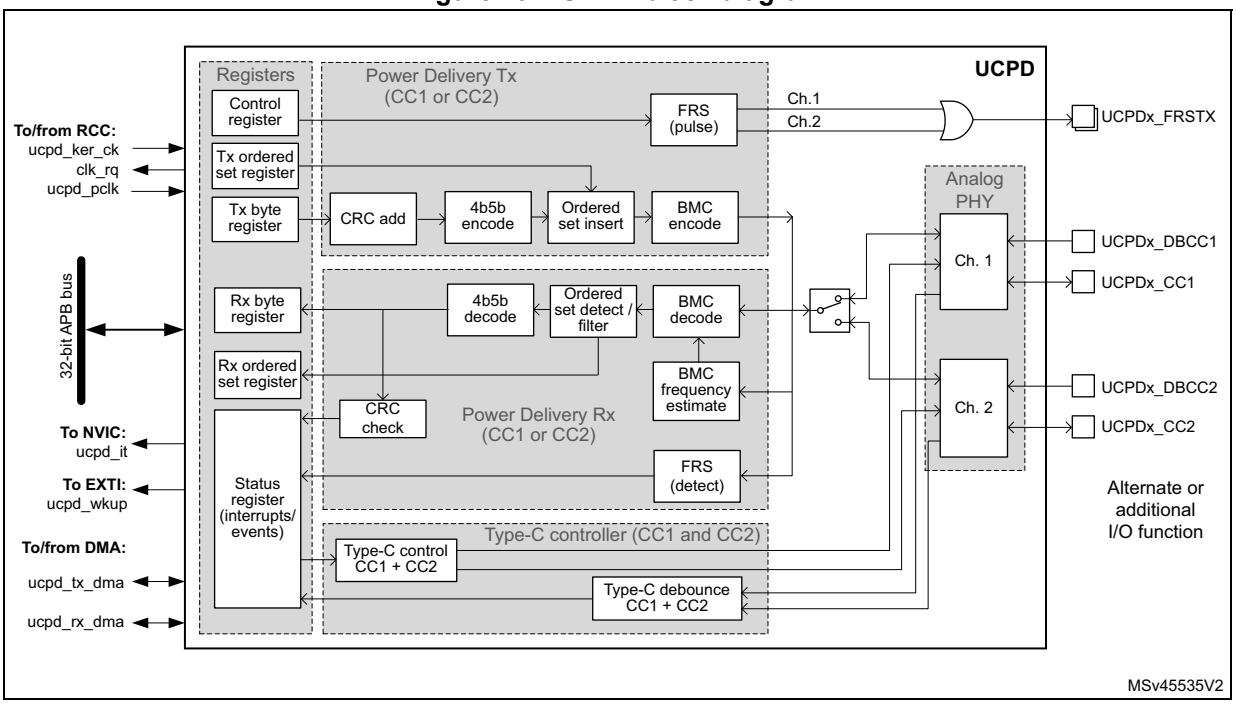
The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).

In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages and FRS signaling, which allows low-power operation.

56.4.1 UCPD block diagram

Figure 791. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 626. UCPD signals on pins

Pin name	Signal type	Description
UCPDx_FRSTX	Output	USB Type-C fast role swap (FRS) signaling control, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the active CC line. A typical application has two such transistors (one per CC line) and reserves a separate I/O to drive either NMOS. Initially, the I/Os are configured as low-driving GPIOs. Upon detecting, through the Type-C state machine, the orientation of the cable attached, which determines the active CC line, the I/O of the active CC line must be set to its UCPDx_FRSTX alternate function and the I/O of the inactive CC line as low-driving GPIO.
UCPDx_CC1	Input/output	USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.

Table 626. UCPD signals on pins (continued)

Pin name	Signal type	Description
UCPDx_CC2	Input/output	USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.
UCPDx_DBCC1	Input	USB Type-C configuration control line 1 dead battery signal, to be routed to the USB Type-C connector CC1 terminal if dead battery support is required.
UCPDx_DBCC2	Input	USB Type-C configuration control line 2 dead battery signal, to be routed to the USB Type-C connector CC2 terminal if dead battery support is required.

The following table lists key internal signals.

Table 627. UCPD internal signals

Internal signal name	Signal type	Description
ucpd_pclk	Input	APB clock for registers
ucpd_ker_ck	Input	Kernel clock
ucpd_tx_dma	Input/Output	Rx DMA acknowledge / request
ucpd_rx_dma	Input/Output	Tx DMA acknowledge / request
ucpd_it	Output	Interrupt request (all interrupts OR-ed) connected to NVIC
ucpd_wkup	Output	Wake-up request connected to EXTI
clk_rq	Output	Clock request connected to RCC

56.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

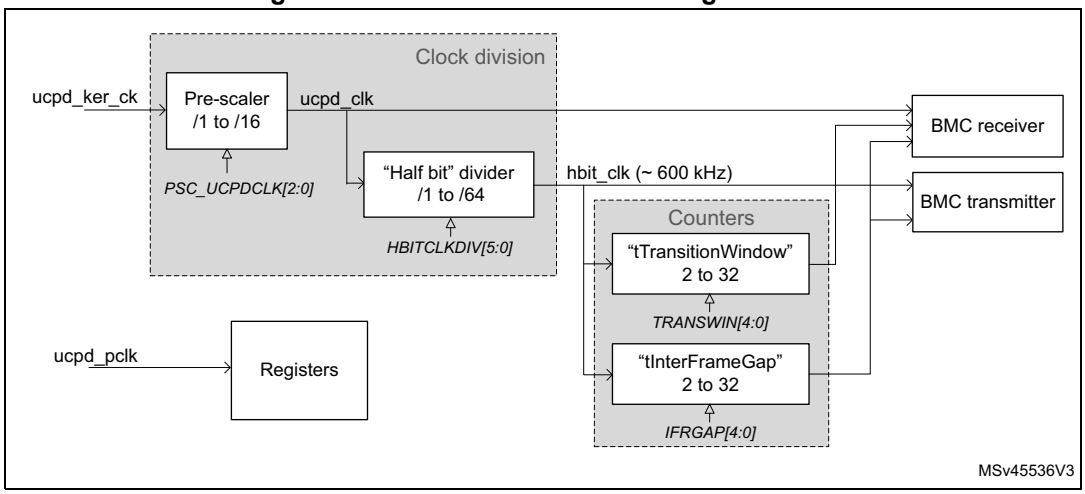
The register section is clocked with the APB clock (ucpd_pclk).

The main functional part of the transmitter is clocked with ucpd_clk clock, pre-scaled from the ucpd_ker_ck clock according to the PSC_UCPDCLK[2:0] bitfield of the UCPD_CFGR1 register. The main functional part of the receiver is clocked with the ucpd_rx_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 6 to 12 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.

Figure 792. Clock division and timing elements



Refer to the USB PD specification in order to set appropriate delays. For *tTransitionWindow* and especially for *tInterFrameGap*, the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.

56.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification.

On the transmitter side its main function is to form packets according to the defined packet format including generally:

- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:

- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

Symbol encoding

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.

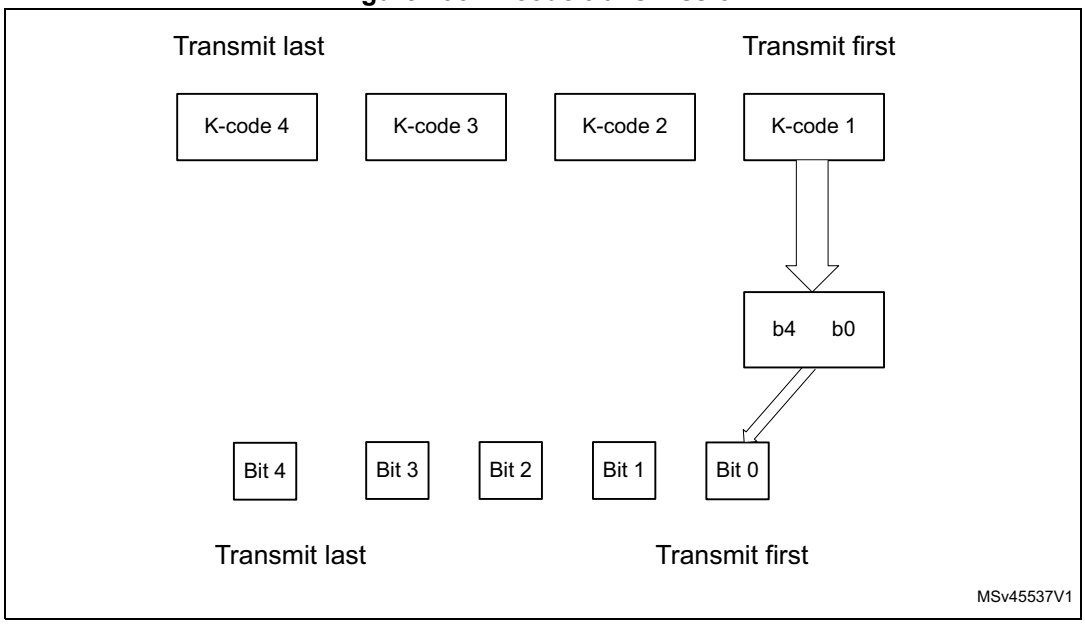
Table 628. 4b5b symbol encoding table

Name	4b	5b	Symbol description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E
F	1111	11101	hex data F
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP
Reserved	Error	00000	Do Not Use
Reserved	Error	00001	Do Not Use
Reserved	Error	00010	Do Not Use
Reserved	Error	00011	Do Not Use
Reserved	Error	00100	Do Not Use
Reserved	Error	00101	Do Not Use
Sync-3	K-code	00110	Startsynch #3
Reserved	Error	01000	Do Not Use
Reserved	Error	01100	Do Not Use
Reserved	Error	10000	Do Not Use
Reserved	Error	11111	Do Not Use

Ordered sets

An ordered set consists of four K-codes as shown in the following figure.

Figure 793. K-code transmission



The following table lists the defined ordered sets, including all possible SOP* sequences.

At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an ongoing Tx message.

Table 629. Ordered sets

Ordered set name	K-code #1	K-code #2	K-code #3	K-code #4
SOP	Sync-1	Sync-1	Sync-1	Sync-2
SOP’	Sync-1	Sync-1	Sync-3	Sync-3
SOP”	Sync-1	Sync-3	Sync-1	Sync-3
Hard Reset	RST-1	RST-1	RST-1	RST-2
Cable Reset	RST-1	Sync-1	RST-1	Sync-3
SOP’_Debug	Sync-1	RST-2	RST-2	Sync-3
SOP”_Debug	Sync-1	RST-2	Sync-3	Sync-2

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

Table 630. Validation of ordered sets

Status	1st code	2nd code	3rd code	4th code
Valid	Corrupt	K-code	K-code	K-code
Valid	K-code	Corrupt	K-code	K-code

Table 630. Validation of ordered sets (continued)

Status	1st code	2nd code	3rd code	4th code
Valid	K-code	K-code	Corrupt	K-code
Valid	K-code	K-code	K-code	Corrupt
Valid (perfect)	K-code	K-code	K-code	K-code
Not valid (example)	K-code	Corrupt	K-code	Corrupt

Bit ordering at transmission

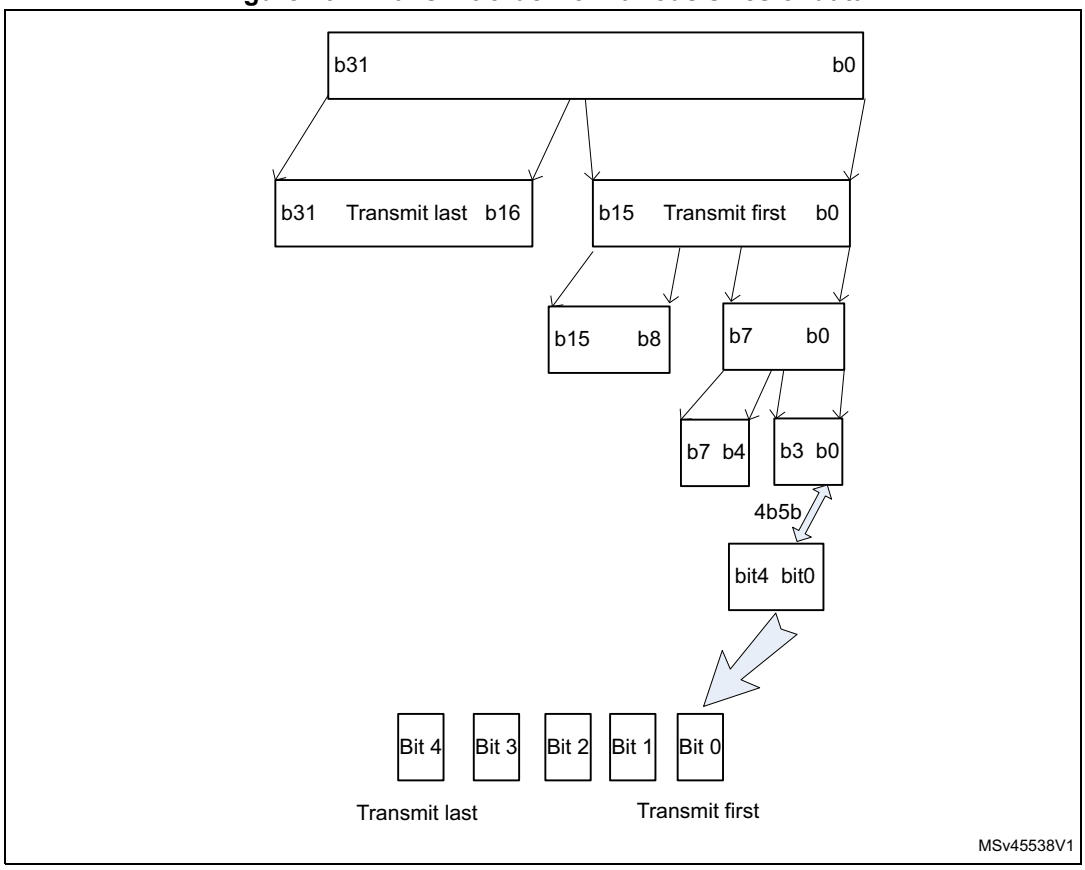
Allowed transmission data units / data sizes are in the following table.

Table 631. Data size

Data unit	Non-encoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20-bits
DWord	32-bits	40-bits

The bit transmission order is shown in the following figure.

Figure 794. Transmit order for various sizes of data

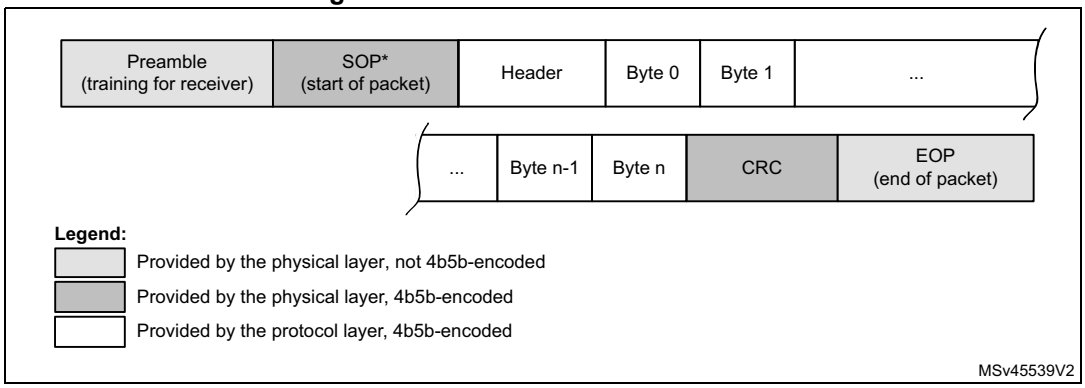


Packet format

Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

Figure 795. Packet format



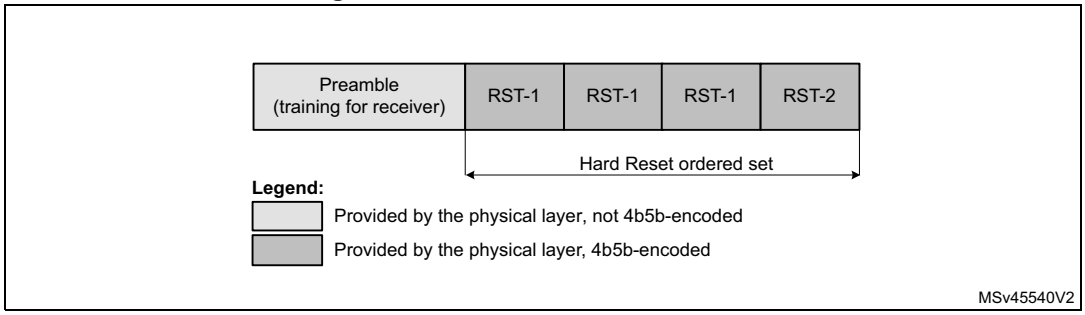
Hard Reset

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an ongoing transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for *tInterFrameGap* time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

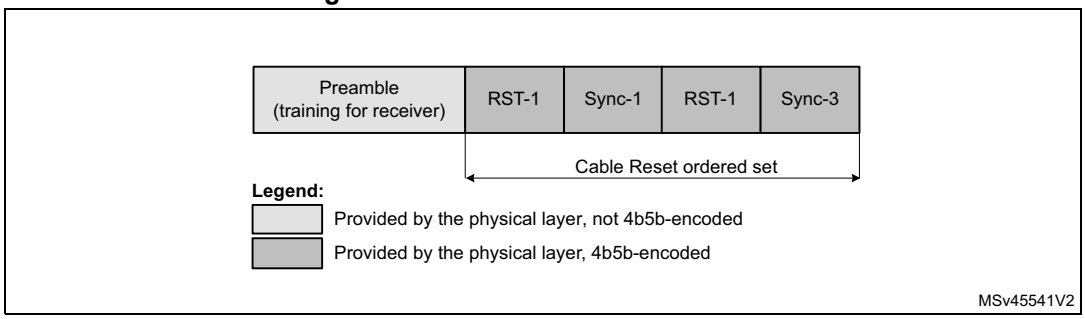
Figure 796. Line format of Hard Reset



Cable Reset

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

Figure 797. Line format of Cable Reset



Collision avoidance

The physical layer respects the *tInterFrameGap* delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three (*nTransitionCount*) transitions within *tTransitionWindow* (12 to 20 µs). The Power Delivery specification revision 3.1 also requires to manage the Rp value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

Physical layer signaling schemes

The bit are signaled with bi-phase mark coding (BMC).

BIST

Depending on the BIST action required by the protocol layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

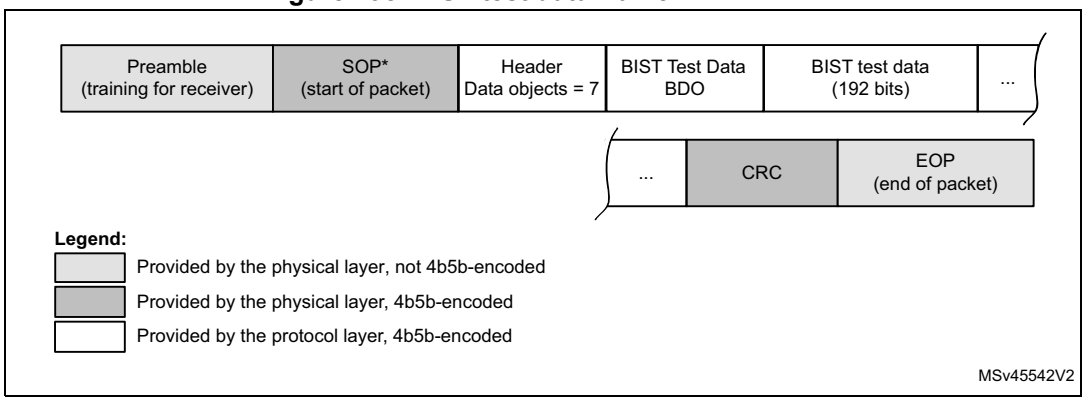
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

BIST test data pattern

The test data pattern is not viewed as a special case in UCPD.

The BIST test data packet frame format is shown in the following figure.

Figure 798. BIST test data frame



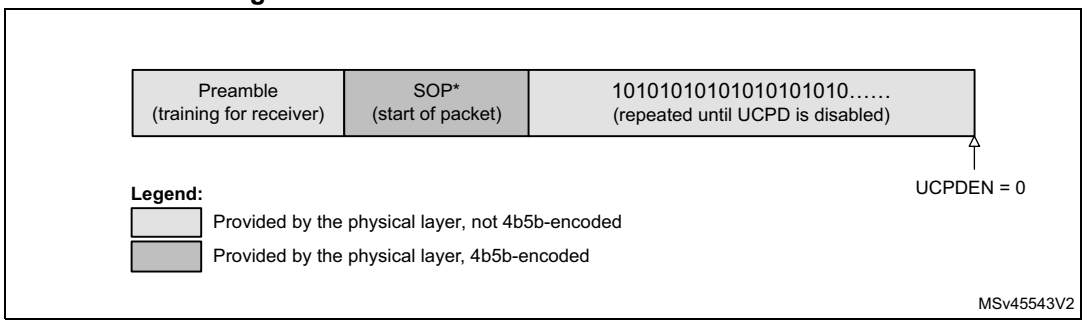
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 795: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block's behavior) that differentiates the general packet from the BIST Test Data packet.

BIST Carrier Mode 2

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

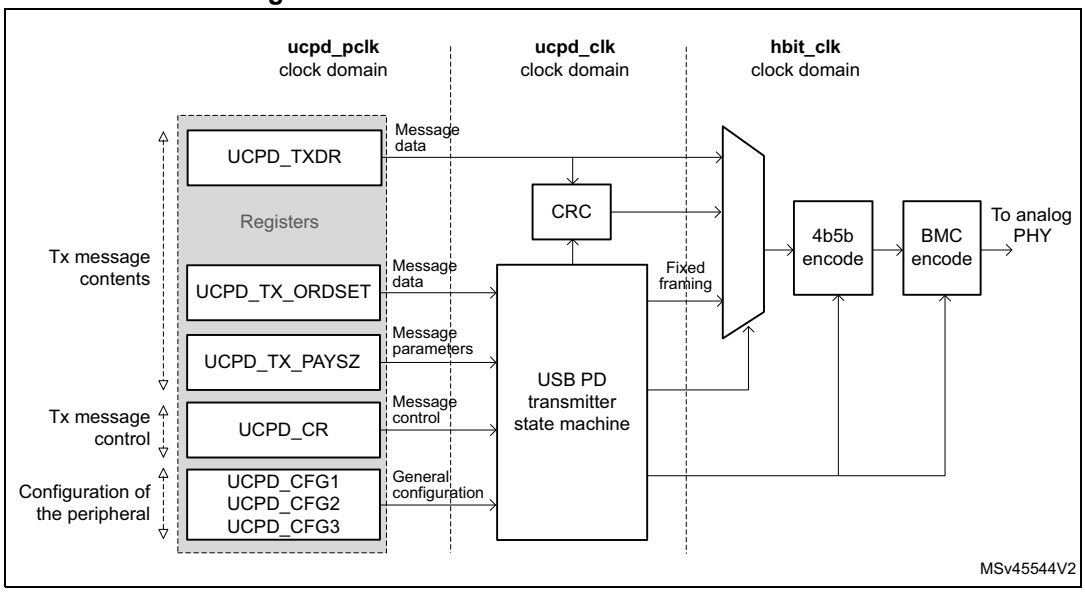
Figure 799. BIST Carrier Mode 2 frame



56.4.4 UCPD BMC transmitter

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.

Figure 800. UCPD BMC transmitter architecture



BMC encoder

The bi-phase mark coding method is defined in the *IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 www.iec.ch* specification.

The half-bit clock hbit_clk is derived from ucpd_clk through a simple divider controlled by the HBITCLKDIV[5:0] bitfield of the UCPD_CFGR1 register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

– *tInterFrameGap*: via IFRGAP (pre-defined value, can be altered)
– *tTransitionWindow*: via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an ongoing Tx message, which (if still in progress) is cleanly terminated by truncating the current sequence and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

Transmitter behavior in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.

56.4.5 UCPD BMC receiver

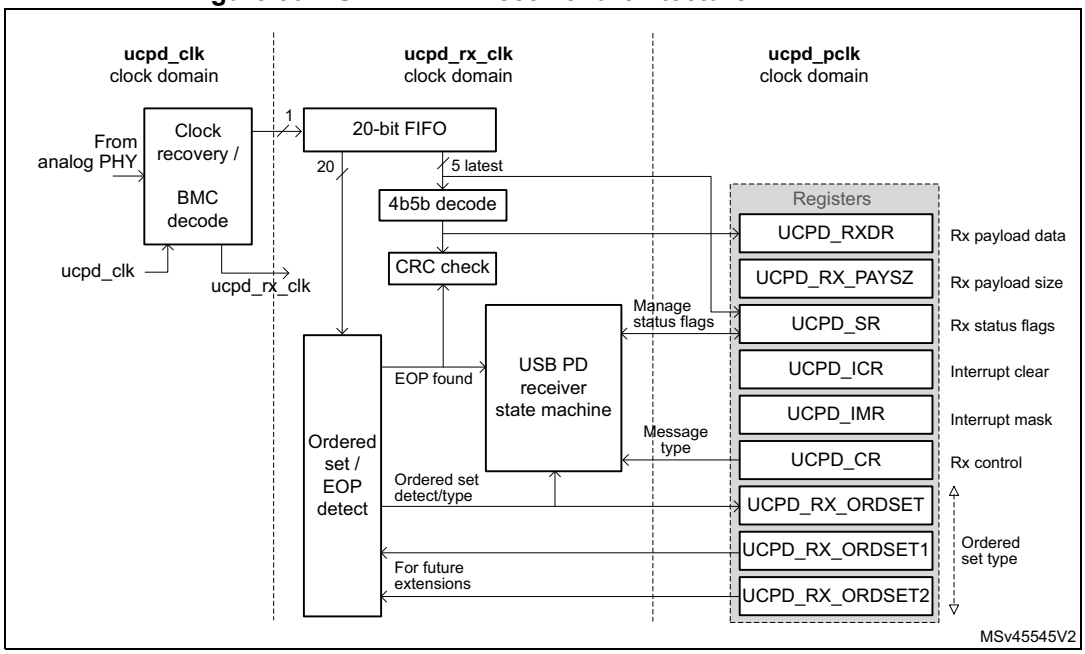
The UCPD BMC receiver performs:

- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

Figure 801. UCPD BMC receiver architecture



CRC checker

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.

Ordered set detection

This function detects the different ordered sets each consisting of four 5-bit K-codes.

Once we are in the preamble we opens a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP* codes (SOP, SOP', and SOP''), but also Hard Reset, Cable Reset, SOP'_Debug, SOP''_Debug, and two extensions defined by registers UCPD_RX_ORDEXT1 and UCPD_RX_ORDEXT2.

EOP detection and Hard Reset exception handling

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection can be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an "early EOP" is detected, the truncated Rx message is immediately discarded.

Truncated or corrupted message exception

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

- the line becomes static for a time significantly longer than one "UI" period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
- the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

Short preamble or incomplete ordered set exception

In the exceptional case of the receiver seeing less that half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

56.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it still presents the "dead battery" Rd, provided that UCPDx_DBCC1 and UCPDx_DBCC2 pins are each connected to UCPDx_CC1 and UCPDx_CC2 pins, respectively.

If dead battery behavior is not required (for example for source only products), then UCPDx_DBCC1 and UCPDx_DBCC2 pins must both be tied to ground.

After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0] before setting the UCPD_DBDIS bit of the PWR_CR3 register to remove dead battery pull-down resistor and allow the values just programmed to take effect.

Use of Standby low-power mode is possible for sinks in the unattached state.

56.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC_VSTATE_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

56.4.8 UCPD fast role swap (FRS)

FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the I/O configured as UCPD_FRSTX (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

FRS detection

FRS monitoring is enabled by setting the bit FRSRXEN, after writing PHYCCSEL that selects the active CC line depending on the cable orientation detected.

56.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle UCPD1_TXDR and UCPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

56.4.10 Wake-up from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.

The events causing the wake-up can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

56.5 UCPD programming sequences

The normal sequence of use of the UCPD unit is:

1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
   – On demand from protocol layer, send Tx message
   – Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

56.5.1 Initialization phase

Use the following sequence for a clean startup:

1. Prepare all initial clock divider values, by writing the UCPD_CFG register.
2. Enable the unit, by setting the UCPDEN bit.
3. Enable the analog Rx filter of either CC line, via the RXAFILTEN bit of the UCPD_CFGR2 register.

56.5.2 Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

Table 632. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx

ANAMODE	ANASUBMODE[1:0]	Notes	TYPEC_VSTATE_CCx[1:0]			
			00	01	10	11
0: Source	00: Disabled	Disabled	N/A			
	01: Default USB Rp	-	vRa[Def]	vRd[Def]	vOPEN[Def]	N/A
	10: 1.5A Rp	-	vRa[1.5]	vRd[1.5]	vOPEN[1.5]	
	11: 3.0A Rp	-	vRa[3.0]	vRd[3.0]	vOPEN[3.0]	
1: Sink	xx	-	vRa	vRd-USB	vRd-1.5	vRd-3.0

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

Note: The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and, when in source mode, determines VCONN generation and

VBUS state (ON/OFF/+voltage level); discharge). UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC line), but the application needs these inputs and controls, to function correctly.

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming, the PHY is effectively off.
2. Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).
3. In the case of no connection, wait for a connection event.
4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.
5. When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHYEVT1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):

Table 633. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

Type-C state	ANAMODE; ANASUBMODE[1:0]	CCENABLE	PHYCCSEL	RDCH	CC[x] VCONNEN(1)	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)	0: [Normal]	00: [neither]	PHYEVT 1: [VRd-3A0]	Wait for sink attach detect ; seen on CC1 [EVT1]
Attachwait. SRC			0 [Rd on CC1]			PHYEVT 2: [VRa]	Attachwait started (100-200 ms) ; now also see the Ra => requesting VCONN
Attached. SRC [VCONN => CC2]	0:Source; 11:Rp3A0 [SinkTxOK]	01: CC2 disable (possible and recommended due to external VCONN switch)			10: [CC2 active]	Timer (100 ms) and no PHYEVTx	Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEVT1
	0:Source; 10:Rp1A5 [SinkTxNG]					SW timers (SinkTxNG)	Source wants to initiate message sequence (SinkTxNG condition set first)
	0:Source; 11:Rp3A0 [SinkTxOK]						Source finished message sequence (SinkTxOK condition afterwards)
						PHYEVT 1: [VOpen-3A0]	Wait for Sink disconnected (Vopen on CC1)
Unattached wait. SRC		11:both enabled	0 (do not care)	1: [discharge]	00: [neither]	> 0.8V detection	Discharge VCONN [CC2] actively [Rdch]; to < 0.8V
Unattached. SRC	0:Source; 11:Rp3A0			0: [Normal]			[Details as first line of table]

1. Two GPIOs to enable VCONN through external load switch components

56.5.3 USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD_TX_ORDSET and UCPD_TX_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD_TXDR register.

This re-iterates until the entire payload of data is transmitted.

Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

Hard Reset transmission

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD_CR register forces the internal state machine to generate the correct sequence. The value of UCPD_TX_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

Use of DMA for transmission

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.

For each message:

- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

56.5.4 USB PD receive

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

The information is recovered by reading:

- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:

- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set

Use of DMA for reception

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.

Whenever a Rx message is expected:

- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

56.5.5 UCPD software trimming

The CC pull-up (Rp) and pull-down (Rd) devices must be trimmed on each part, to meet the required accuracy. The trimming values are saved in the non-volatile memory.

To trim the CC pull-up and pull-down devices by software, apply the following procedure:

1. Retrieve the trim values from the non-volatile memory (refer to *Table 625: UCPD software trim data*)
2. At initialization, write the trim values to the UCPD_CFGR3 register bitfields as follows:
   - 3A0_CC1[3:0] to TRIM_CC1_RP[3:0]
   - 3A0_CC2[3:0] to TRIM_CC2_RP[3:0]
   - Rd_CC1[3:0] to TRIM_CC1_RD[3:0]
   - Rd_CC2[3:0] to TRIM_CC2_RD[3:0]
3. At each setting of ANASUBMODE to 1A5 or 3A0, respectively, write the trimming values to the UCPD_CFGR3 register bitfields as follows:
   - 1A5_CC1[3:0] or 3A0_CC1[3:0], respectively, to TRIM_CC1_RP[3:0]
   - 1A5_CC2[3:0] or 3A0_CC2[3:0], respectively, to TRIM_CC2_RP[3:0]

56.6 UCPD low-power modes

A summary of low-power modes is shown below in *Table 634: Effect of low power modes on the UCPD*.

Table 634. Effect of low power modes on the UCPD

Mode	Description
Sleep	No effect
Stop	Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.
Standby	UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.
Unpowered	Dead battery pull-downs remain active.

The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wake-up from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which can later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which can potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which can potentially generate an FRS signaling detection event (that is, FRSEVT) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

56.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

Table 635. UCPD interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
FRS detection	FRSEVT	Set FRSEVTCF	FRSEVTIE
Type C voltage level change on CC2	TYPECEVT2	Set TYPECEVT2CF	TYPECEVT2IE
Type C voltage level change on CC1	TYPECEVT1	Set TYPECEVT1CF	TYPECEVT1IE
Rx message received	RXMSGEND	Set RXMSGENDCF	RXMSGENDIE
Rx data overflow	RXOVR	Set RXOVRCF	RXOVR
Rx Hard Reset detected	RXHRSTDET	Set RXHRSTDETCF	RXHRSTDETIE
Rx ordered set (4 K-codes) detected	RXORDDET	Set RXORDDETCF	RXORDDETIE
Receive data register not empty	RXNE	Read data in UCPD_RXDR	RXNEIE
Tx data underrun	TXUND	Set TXUNDCF	TXUNDIE
Hard Reset sent	HRSTSENT	Set HRSTSENTCF	HRSTSENTIE
Hard Reset discarded	HRSTDISC	Set HRSTDISCCF	HRSTDISCIE
Transmit message aborted	TXMSGABT	Set TXMSGABTCF	TXMSGABTIE

Table 635. UCPD interrupt requests (continued)

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Transmit message sent	TXMSGSENT	Set TXMSGSENTCF	TXMSGSENTIE
Transmit message discarded	TXMSGDISC	Set TXMSGDISCCF	TXMSGDISCIE
Transmit data required	TXIS	Write data to the UCPD_TXDR register	TXISIE

When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register.

Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD_ICR register.

56.8 UCPD registers

56.8.1 UCPD configuration register 1 (UCPD_CFGR1)

Address offset: 0x000

Reset value: 0x0000 0000

General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

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

Bit 31 **UCPDEN**: UCPD peripheral enable

General enable of the UCPD peripheral.

0: Disable

1: Enable

Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

Bit 30 **RXDMAEN**: Reception DMA mode enable

When set, the bit enables DMA mode for reception.

0: Disable

1: Enable

Bit 29 **TXDMAEN**: Transmission DMA mode enable

When set, the bit enables DMA mode for transmission.

0: Disable

1: Enable

Bits 28:20 **RXORDSETEN[8:0]**: Receiver ordered set enable

The bitfield determines the types of ordered sets that the receiver must detect. When set/cleared, each bit enables/disables a specific function:
0bXXXXXXXX1: SOP detect enabled
0bXXXXXXX1X: SOP' detect enabled
0bXXXXXX1XX: SOP'' detect enabled
0bXXXXX1XXX: Hard Reset detect enabled
0bXXXX1XXXX: Cable Detect reset enabled
0bXXX1XXXXX: SOP'_Debug enabled
0bXX1XXXXXX: SOP''_Debug enabled
0bX1XXXXXXX: SOP extension#1 enabled
0b1XXXXXXXX: SOP extension#2 enabled

Bits 19:17 **PSC_UCPDCLK[2:0]**: Pre-scaler division ratio for generating ucpd_clk

The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD peripheral clock (ucpd_clk).
0x0: 1 (bypass)
0x1: 2
0x2: 4
0x3: 8
0x4: 16
It is recommended to use the pre-scaler so as to set the ucpd_clk frequency in the range from 6 to 9 MHz.

Bit 16 Reserved, must be kept at reset value.

Bits 15:11 **TRANSWIN[4:0]**: Transition window duration

The bitfield determines the division ratio (the bitfield value minus one) of a hbit_clk divider producing *tTransitionWindow* interval.
0x00: Not supported
0x01: 2
0x09: 10 (recommended)
0x1F: 32
Set a value that produces an interval of 12 to 20 us, taking into account the ucpd_clk frequency and the HBITCLKDIV[5:0] bitfield setting.

Bits 10:6 **IFRGAP[4:0]**: Division ratio for producing inter-frame gap timer clock

The bitfield determines the division ratio (the bitfield value minus one) of a ucpd_clk divider producing inter-frame gap timer clock (*tInterFrameGap*).
0x00: Not supported
0x01: 2
0x0D: 14
0x0E: 15
0x0F: 16
0x1F: 32
The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value. The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios above 15 for Tx clock above nominal.

Bits 5:0 **HBITCLKDIV[5:0]**: Division ratio for producing half-bit clock

The bitfield determines the division ratio (the bitfield value plus one) of a ucpd_clk divider producing half-bit clock (hbit_clk).
0x00: 1 (bypass)
0x1A: 27
0x3F: 64

56.8.2 UCPD configuration register 2 (UCPD_CFGR2)

Address offset: 0x004

Reset value: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
							rw					rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RXAFILTEN**: Rx analog filter enable
Setting the bit enables the Rx analog filter required for optimum Power Delivery reception.
0: Disable
1: Enable

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **WUPEN**: Wake-up from Stop mode enable
Setting the bit enables the UCPD_ASYNC_INT signal.
0: Disable
1: Enable

Bit 2 **FORCECLK**: Force ClkReq clock request
0: Do not force clock request
1: Force clock request

Bit 1 **RXFILT2N3**: BMC decoder Rx pre-filter sampling method
Number of consistent consecutive samples before confirming a new value.
0: 3 samples
1: 2 samples

Bit 0 **RXFILTDIS**: BMC decoder Rx pre-filter enable
0: Enable
1: Disable
The sampling clock is that of the receiver (that is, after pre-scaler).

56.8.3 UCPD configuration register 3 (UCPD_CFGR3)

Address offset: 0x008

Reset value: 0x0000 0000

Configuration of UCPD trimming of the CC pull-up and pull-down devices. The trimming is managed by hardware until the first software write into this register.

The register is reserved (must not be written) for devices that support the fully automatic trimming. Refer to *Section 56.3: UCPD implementation*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TRIM_CC2_RP[3:0]				Res.	Res.	Res.	Res.	Res.	TRIM_CC2_RD[3:0]			
			rw	rw	rw	rw						rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TRIM_CC1_RP[3:0]				Res.	Res.	Res.	Res.	Res.	TRIM_CC1_RD[3:0]			
			rw	rw	rw	rw						rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **TRIM_CC2_RP[3:0]**: SW trim value for Rp current sources on the CC2 line

Bits 24:20 Reserved, must be kept at reset value.

Bits 19:16 **TRIM_CC2_RD[3:0]**: SW trim value for Rd resistor on the CC2 line

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:9 **TRIM_CC1_RP[3:0]**: SW trim value for Rp current sources on the CC1 line

Bits 8:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIM_CC1_RD[3:0]**: SW trim value for Rd resistor on the CC1 line

56.8.4 UCPD control register (UCPD_CR)

Address offset: 0x00C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN
										rw	rw		rw	rs	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw	rw

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

Bit 21 **CC2TCDIS**: CC2 Type-C detector disable

The bit disables the Type-C detector on the CC2 line.
0: Enable
1: Disable
When enabled, the Type-C detector for CC2 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 20 **CC1TCDIS**: CC1 Type-C detector disable

The bit disables the Type-C detector on the CC1 line.
0: Enable
1: Disable
When enabled, the Type-C detector for CC1 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RDCH**: Rdch condition drive

The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus associated with VCONN), by remaining set during the source-only *UnattachedWait.SRC* state, to respect the Type-C state. Refer to *"USB Type-C ECN for Source VCONN Discharge"*. The CCENABLE[1:0] bitfield must be set accordingly, too.

0: No effect

1: Rdch condition drive

Bit 17 **FRSTX**: FRS Tx signaling enable.

Setting the bit enables FRS Tx signaling.

0: No effect

1: Enable

The bit is cleared by hardware after a delay respecting the USB Power Delivery specification Revision 3.1.

Bit 16 **FRSRXEN**: FRS event detection enable

Setting the bit enables FRS Rx event (FRSEVT) detection on the CC line selected through the PHYCCSEL bit. 0: Disable

1: Enable

Clear the bit when the device is attached to an FRS-incapable source/sink.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **CCENABLE[1:0]**: CC line enable

This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to ANAMODE and ANASUBMODE[1:0] setting.

0x0: Disable both PHYs

0x1: Enable CC1 PHY

0x2: Enable CC2 PHY

0x3: Enable CC1 and CC2 PHY

A single line PHY can be enabled when, for example, the other line is driven by VCONN via an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.

Bit 9 **ANAMODE**: Analog PHY operating mode

0: Source

1: Sink

The use of CC1 and CC2 depends on CCENABLE. Refer to *Table 632: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield in conjunction with ANASUBMODE[1:0].

Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode

Refer to *Table 632: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield.

Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling

0: Use CC1 IO for Power Delivery communication

1: Use CC2 IO for Power Delivery communication

The selection depends on the cable orientation as discovered at attach.

Bit 5 **PHYRXEN**: USB Power Delivery receiver enable

0: Disable

1: Enable

Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode

Determines the mode of the receiver.

0: Normal receive mode

1: BIST receive mode (BIST test data mode)

When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset

0: No effect

1: Start Tx Hard Reset message

The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet

0: No effect

1: Start Tx packet transmission

The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet

Writing the bitfield triggers the action as follows, depending on the value:

0x0: Transmission of Tx packet previously defined in other registers

0x1: Cable Reset sequence

0x2: BIST test sequence (BIST Carrier Mode 2)

Others: invalid

From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).

56.8.5 UCPD interrupt mask register (UCPD_IMR)

Address offset: 0x010

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTIE	Res.	Res.	Res.	Res.
											r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTIE**: FRSEVT interrupt enable
0: Disable
1: Enable

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable
0: Disable
1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable
0: Disable
1: Enable

Bit 11 **RXOVRIE**: RXOVR interrupt enable
0: Disable
1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable
0: Disable
1: Enable

Bit 9 **RXORDDETIE**: RXORDDET interrupt enable
0: Disable
1: Enable

Bit 8 **RXNEIE**: RXNE interrupt enable
0: Disable
1: Enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUNDIE**: TXUND interrupt enable
0: Disable
1: Enable

Bit 5 **HRSTSENTIE**: HRSTSENT interrupt enable
0: Disable
1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable
0: Disable
1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable
0: Disable
1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable
0: Disable
1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable
0: Disable
1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable
0: Disable
1: Enable

56.8.6 UCPD status register (UCPD_SR)

Address offset: 0x014

Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD_IMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]	
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
r	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVT**: FRS detection event
The flag is cleared by setting the FRSEVTCF bit.
0: No new event
1: New FRS receive event occurred

Bits 19:18 **TYPEC_VSTATE_CC2[1:0]**: CC2 line voltage level
The status bitfield indicates the voltage level on the CC2 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16 **TYPEC_VSTATE_CC1[1:0]**:
The status bitfield indicates the voltage level on the CC1 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bit 15 **TYPECEVT2**: Type-C voltage level event on CC2 line
The flag indicates a change of the TYPEC_VSTATE_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.
0: No new event
1: A new Type-C event

Bit 14 **TYPECEVT1**: Type-C voltage level event on CC1 line

The flag indicates a change of the TYPEC_VSTATE_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.

0: No new event

1: A new Type-C event

Bit 13 **RXERR**: Receive message error

The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.

0: No error detected

1: Error(s) detected

Bit 12 **RXMSGEND**: Rx message received

The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.

0: No new Rx message received

1: A new Rx message received

The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.

Bit 11 **RXOVR**: Rx data overflow detection

The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVRCF bit.

0: No overflow

1: Overflow

The buffer overflow can occur if the received data are not read fast enough.

Bit 10 **RXHRSTDET**: Rx Hard Reset receipt detection

The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDETCF bit.

0: Hard Reset not received

1: Hard Reset received

Bit 9 **RXORDDET**: Rx ordered set (4 K-codes) detection

The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD_RX_ORDSET register. It is cleared by setting the RXORDDETCF bit.

0: No ordered set detected

1: A new ordered set detected

Bit 8 **RXNE**: Receive data register not empty detection

The flag indicates that the UCPD_RXDR register is not empty. It is automatically cleared upon reading UCPD_RXDR.

0: Rx data register empty

1: Rx data register not empty

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUND**: Tx data underrun detection

The flag indicates that the Tx data register (UCPD_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.

0: No Tx data underrun detected

1: Tx data underrun detected

Bit 5 **HRSTSENT**: Hard Reset message sent

The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENTCF bit.

0: No Hard Reset message sent

1: Hard Reset message sent

Bit 4 **HRSTDISC**: Hard Reset discarded

The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.

0: No Hard Reset discarded

1: Hard Reset discarded

Bit 3 **TXMSGABT**: Transmit message abort

The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.

0: No transmit message abort

1: Transmit message abort

Bit 2 **TXMSGSENT**: Message transmission completed

The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSENTCF bit.

0: No Tx message completed

1: Tx message completed

In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.

Bit 1 **TXMSGDISC**: Message transmission discarded

The flag indicates that a message transmission was dropped. The flag is cleared by setting the TXMSGDISCCF bit.

0: No Tx message discarded

1: Tx message discarded

Transmission of a message can be dropped if there is a concurrent receive in progress or at excessive noise on the line. After a Tx message is discarded, the flag is only raised when the CC line becomes idle.

Bit 0 **TXIS**: Transmit interrupt status

The flag indicates that the UCPD_TXDR register is empty and new data write is required (as the amount of data sent has not reached the payload size defined in the TXPAYSZ bitfield). The flag is cleared with the data write into the UCPD_TXDR register.

0: New Tx data write not required

1: New Tx data write required

56.8.7 UCPD interrupt clear register (UCPD_ICR)

Address offset: 0x018

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.
											w				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
w	w		w	w	w	w			w	w	w	w	w	w	

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **FRSEVTCF**: FRS event flag (FRSEVT) clear
Setting the bit clears the FRSEVT flag in the UCPD_SR register.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2CF**: Type-C CC2 line event flag (TYPECEVT2) clear
Setting the bit clears the TYPECEVT2 flag in the UCPD_SR register

Bit 14 **TYPECEVT1CF**: Type-C CC1 event flag (TYPECEVT1) clear
Setting the bit clears the TYPECEVT1 flag in the UCPD_SR register

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear
Setting the bit clears the RXMSGEND flag in the UCPD_SR register.

Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear
Setting the bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detect flag (RXHRSTDET) clear
Setting the bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear
Setting the bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear
Setting the bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENTCF**: Hard reset send flag (HRSTSENT) clear
Setting the bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear
Setting the bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear
Setting the bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENTCF**: Tx message send flag (TXMSGSENT) clear
Setting the bit clears the TXMSGSENT flag in the UCPD_SR register.

Bit 1 **TXMSGDISCCF**: Tx message discard flag (TXMSGDISC) clear
Setting the bit clears the TXMSGDISC flag in the UCPD_SR register.

Bit 0 Reserved, must be kept at reset value.

56.8.8 UCPD Tx ordered set type register (UCPD_TX_ORDSETR)

Address offset: 0x01C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **TXORDSET[19:0]**: Ordered set to transmit
The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.

56.8.9 UCPD Tx payload size register (UCPD_TX_PAYSZR)

Address offset: 0x020

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

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

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

Bits 9:0 **TXPAYSZ[9:0]**: Payload size yet to transmit

The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission.

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

56.8.10 UCPD Tx data register (UCPD_TXDR)

Address offset: 0x024

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

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

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

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

Bits 7:0 **TXDATA[7:0]**: Data byte to transmit

56.8.11 UCPD Rx ordered set register (UCPD_RX_ORDSETR)

Address offset: 0x028

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP 3OF4	RXORDSET[2:0]		
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **RXSOPKINVALID[2:0]**:
The bitfield is for debug purposes only.
0x0: No K-code corrupted
0x1: First K-code corrupted
0x2: Second K-code corrupted
0x3: Third K-code corrupted
0x4: Fourth K-code corrupted
Others: Invalid

Bit 3 **RXSOP3OF4**:
The bit indicates the number of correct K-codes. For debug purposes only.
0: 4 correct K-codes out of 4
1: 3 correct K-codes out of 4

Bits 2:0 **RXORDSET[2:0]**: Rx ordered set code detected
0x0: SOP code detected in receiver
0x1: SOP' code detected in receiver
0x2: SOP'' code detected in receiver
0x3: SOP'_Debug detected in receiver
0x4: SOP''_Debug detected in receiver
0x5: Cable Reset detected in receiver
0x6: SOP extension#1 detected in receiver
0x7: SOP extension#2 detected in receiver

56.8.12 UCPD Rx payload size register (UCPD_RX_PAYSZR)

Address offset: 0x02C

Reset value: 0x0000 0000

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

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

Bits 9:0 **RXPAYSZ[9:0]**: Rx payload size received
This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).
0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)
The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

56.8.13 UCPD receive data register (UCPD_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

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

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

Bits 7:0 **RXDATA[7:0]**: Data byte received

56.8.14 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1)

Address offset: 0x034

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX1[19:0]**: Ordered set 1 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

56.8.15 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2)

Address offset: 0x038

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX2[19:0]**: Ordered set 2 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

56.8.16 UCPD register map

Table 636. UCPD register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	UCPD_CFGR1	UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_UCPDCLK[2:0]			Res.	TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	UCPD_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXAFILTEN	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
	Reset value																								0					0	0	0	0
0x008	UCPD_CFGR3	Res.	Res.	Res.	TRIM_CC2_RP[3:0]				Res.	Res.	Res.	Res.	Res.	TRIM_CC2_RD[3:0]				Res.	Res.	Res.	TRIM_CC1_RP[3:0]			Res.	Res.	Res.	Res.	Res.	Res.	TRIM_CC1_RD[3:0]			
	Reset value				0	0	0	0						0	0	0	0				0	0	0	0						0	0	0	0

Table 636. UCPD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	UCPD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	FRSRXEN	Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
	Reset value											0	0		0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x010	UCPD_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTIE	Res.	Res.	Res.	Res.	TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
	Reset value												0					0	0		0	0	0	0	0		0	0	0	0	0	0	0
0x014	UCPD_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVT	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]		TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x018	UCPD_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRSEVTCF	Res.	Res.	Res.	Res.	TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
	Reset value												0					0	0		0	0	0	0			0	0	0	0	0	0	
0x01C	UCPD_TX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	UCPD_TX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	UCPD_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x028	UCPD_RX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
	Reset value																										0	0	0	0	0	0	0
0x02C	UCPD_RX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0

Table 636. UCPD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	UCPD_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x034	UCPD_RX_ORDEXTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	UCPD_RX_ORDEXTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C - 0x3FF	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

57 Ethernet (ETH): media access control (MAC) with DMA controller

57.1 Ethernet introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The Ethernet peripheral enables to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard.

The peripheral is configurable to meet the needs of a large variety of consumer and industrial applications.

57.2 Ethernet main features

The Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

57.2.1 Standard compliance

The Ethernet peripheral is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC and media independent interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master and AHB slave ports
- RMII specification version 1.2 from RMII consortium

57.2.2 MAC features

MAC Tx and Rx common features

- Separate transmission, reception, and control interfaces to the application
- 10, 100 Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD protocol support
  - Flow control using backpressure (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)
- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII PHYs

- 32-bit data transfer interface on the application side
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Network statistics with RMON or MIB counters (partial support of RFC2819/RFC2665)
- Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping are supported in Tx direction.
- Flexibility to control pulse-per-second (PPS) output signal (eth_ptp_pps_out and ETH_PPS_OUT)
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

MAC Tx features

- Preamble and start-of-frame data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or Jumbo Ethernet packets of up to 16 Kbytes
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in Full-duplex mode)
- Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in Full-duplex mode
- Insert, replace, or delete queue/channel-based VLAN tags

MAC Rx features

- Automatic Pad and CRC stripping options
- Option to disable automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - Four 48-bit perfect (DA) address filters with masks for each byte
  - Four 48-bit SA address comparison check with masks for each byte
  - 64 bit Hash filter for multicast and unicast (DA) addresses
- Option to pass all multicast addressed packets
- Promiscuous mode to pass all packets without any filtering for network monitoring
- Pass all incoming packets (as per filter) with a status report

- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering based either on the outer or inner VLAN tag
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- Detection of remote wake-up packets and AMD magic packets
- Optional forwarding of received Pause packets to the application (in Full-duplex mode)
- Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN tags and providing the tags in the status

57.2.3 Transaction layer (MTL) features

MTL Tx and Rx Common Features

- 32-bit Transaction Layer block (bridges the application and the MAC)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)

MTL Tx features

- 2048-byte Transmit FIFO with programmable threshold capability
- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Automatic retransmission of collision packets in Half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum on frames transmitted in Store-and-forward mode
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Packet-level control for
  - VLAN tag insertion or replacement
  - Ethernet source address insertion
  - Layer 3/Layer 4 checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control

MTL Rx features

- 2048-byte Receive FIFO with configurable threshold
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO

57.2.4 DMA block features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see *Section 57.11.2: Ethernet DMA registers*) to control the DMA operations. The DMA block supports the following features:

- 32-bit data transfers
- Separate DMA in Transmit path and receive paths
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host control (AHB) access and host data interface
- Tx DMA channel with TCP segmentation offload (TSO) feature enabled
- Programmable control for Transmit Descriptor posted writes to improve the throughput

57.2.5 Bus interface features

AHB master interface

The AHB master interface features are the following:

- Interfaces with the application through AHB
- 32-bit data on the AHB master port
- Split, Retry, and Error AHB responses
- AHB 1-Kbyte boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)

The AHB master interface does not generate the following:

- Wrap burst
- Locked or protected transfers

AHB slave interface

The AHB slave interface supports the following features:

- Interfaces with the application through AHB
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:

- Split
- Retry
- Error

57.3 Ethernet pins and internal signals

Table 637 lists the Ethernet inputs and output signals connected to package pins or balls. Active pins depend on the PHY type selected (MII or RMII) and on the device configuration.

Table 638 shows the internal Ethernet signals.

Table 637. Ethernet peripheral pins

Port name	Digital port type	Description
ETH_COL	Input	Collision detection signal, MII only.
ETH_CRS	input	Carrier sense signal, MII only
ETH_REF_CLK	Input	RMII reference clock
ETH_RX_CLK	Input	MII timing reference for Rx data transfers
ETH_RXD[3:0]	Input	Receive data. 4 pins for MII, 2 for RMII.
ETH_RX_DV	Input	Receive data valid
ETH_CRS_DV	Input	RMII: Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
ETH_RX_ER	Input	Receive error
ETH_TX_CLK	Input	MII timing reference for Tx data transfers
ETH_TXD[3:0]	Output	Transmit data. 4 pins for MII, 2 for RMII.
ETH_TX_EN	Output	Transmit data enable
ETH_TX_ER	Output	Transmit error
ETH_MDC	Output	Management data clock
ETH_MDIO	Input/output	Management data
ETH_PHY_INTN	Input	PHY interrupt

Table 637. Ethernet peripheral pins (continued)

Port name	Digital port type	Description
ETH_PPS_OUT	Output	PTP pulse-per-second output
ETH_PTP_AUX_TS(1)	Input	Trigger input for the auxiliary snapshots of the PTP system time

1. This pin is not available on all devices.

Table 638. Ethernet internal input/output signals

Signal name	Signal type	Description
eth_hclk	Digital input	AHB clock
eth_sbd_intr_it	Digital output	Main Ethernet interrupt
lpi_intr_o	Digital output	Sideband signal generated when the transmitter or receiver enters or exits the LPI state.
pmt_intr_o	Digital output	Sideband signal generated when a valid remote wake-up packet is received
eth_tx_clk	Digital input	Tx kernel clock
eth_rx_clk	Digital input	Rx kernel clock
eth_rmii_ref_clk	Digital input	RMII reference kernel clock
eth_ptp_pps_out	Digital output	PTP pulse-per-second signal
mac_speed_o[1:0]	Digital output	MAC speed information used by the RCC
clk_ptp_ref_i	Digital input	PTP reference clock input
eth_ptp_trig[3:0]	Digital input	Trigger input for auxiliary snapshots of the PTP system time

57.4 Ethernet architecture

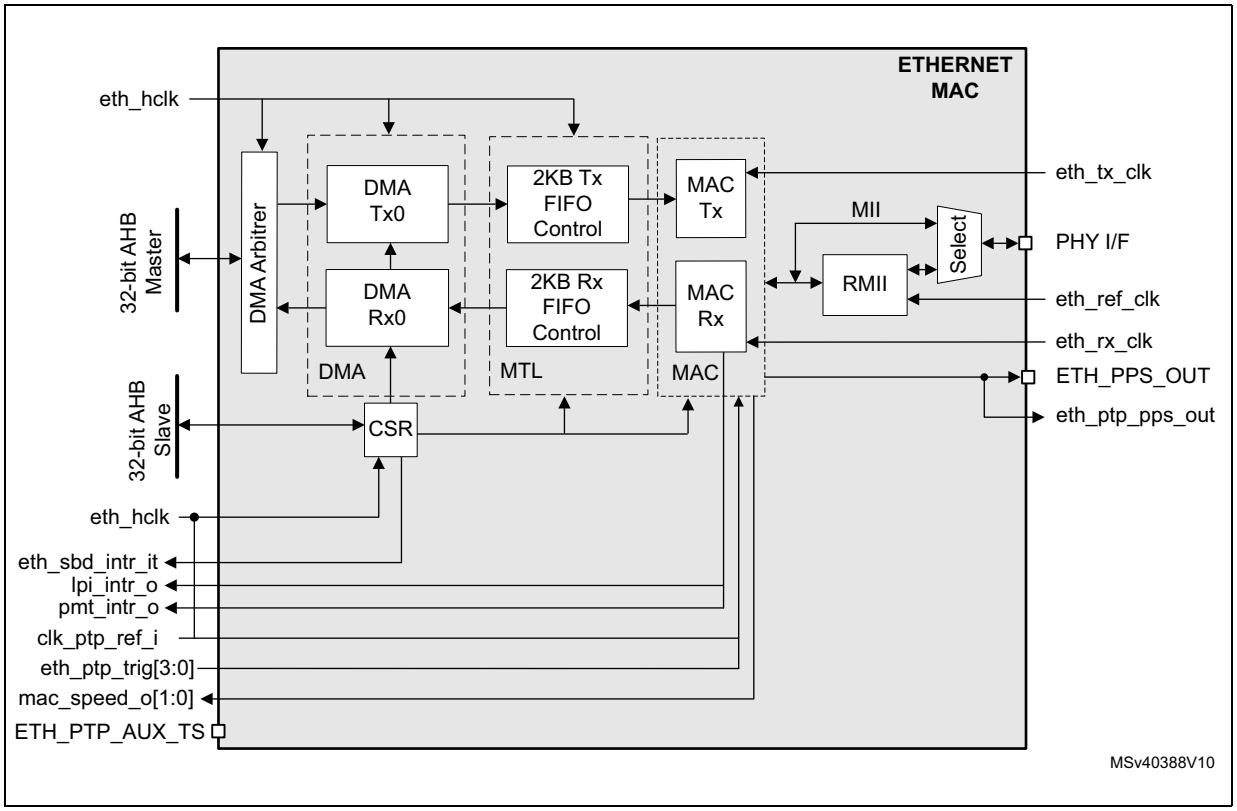
The Ethernet peripheral is composed of 4 main functional modules:

- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)

  This is the logical DMA module with one physical channel for reception and one for transmission. It controls the data transfers between MAC and system memory through the AMBA AHB 32-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The MAC transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII PHY Media Independent Interfaces.

Figure 802. Ethernet high-level block diagram



1. For a definition of the internal signals, refer to *Table 638*.
2. Refer to RCC chapter "Clock distribution for Ethernet" for a detailed description of the Ethernet clock architecture.
3. The ETH_PTP_AUX_TS pin is not available on all devices.

57.4.1 DMA controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The base address of each list is written to the respective Tx and Rx registers: *Channel Tx descriptor list address register (ETH_DMACTXDLAR)*) and *Channel Rx descriptor list address register (ETH_DMACRXDLAR)*.

The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, *Channel Tx descriptor ring length register (ETH_DMACTXRLR)* and *Channel Rx descriptor ring length register (ETH_DMACRXRLR)*.

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

Descriptors are specified in *Section 57.10: Descriptors*.

DMA arbitration

The DMA module incorporates an arbiter that performs the arbitration between the Tx and Rx channels accesses from the AHB master interface. The following two types of arbitrations are supported and can be selected through *DMA mode register (ETH_DMAMR)*:

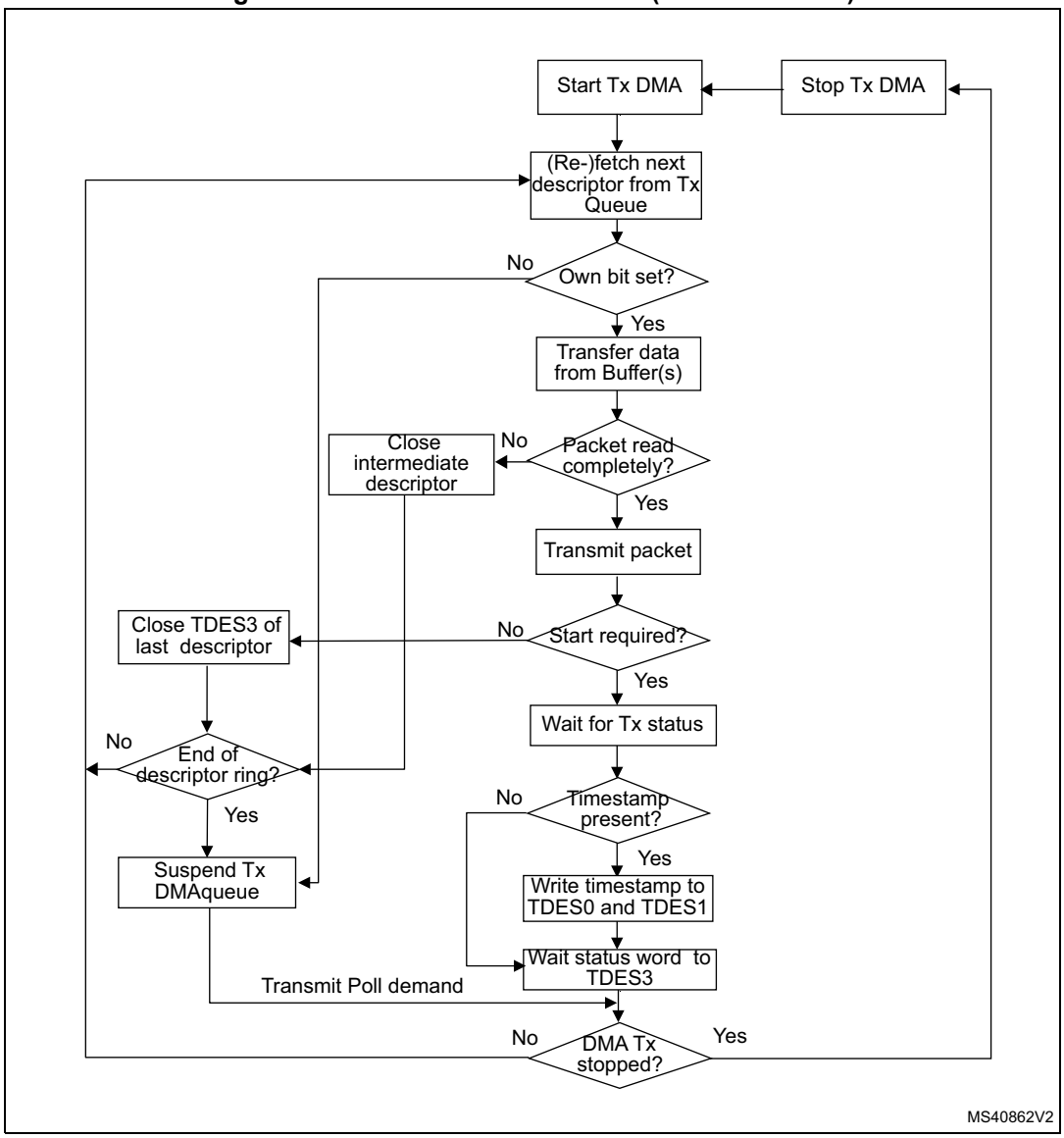
- Round-robin arbitration**:** the arbiter allocates the data bus between Rx and Tx in ratio set by Bits [14:12] of ETH_DMAMR.
- Fixed-priority arbitration**:** by default Rx DMA always gets priority over Tx DMA for data access. Setting bit 11 of ETH_DMAMR register gives priority to the Tx DMA.

DMA transmission in default mode

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application shifts the descriptor tail pointer offset value of the Transmit channel.
3. The DMA fetches the descriptor from the application memory.
4. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   – The descriptor is flagged as owned by the application (TDES3 [31] = 0).
   – The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   – An error condition occurs.
5. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
6. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
7. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.
8. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.
9. Bit 0 of *Channel status register (ETH_DMACSR)*) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
10. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the *Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)* when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.

Figure 803. DMA transmission flow (standard mode)



DMA transmission in OSP (Operate on Second Packet) mode

In Run state, if bit 4 is set in the *Channel transmit control register (ETH_DMACTXCR)*), the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

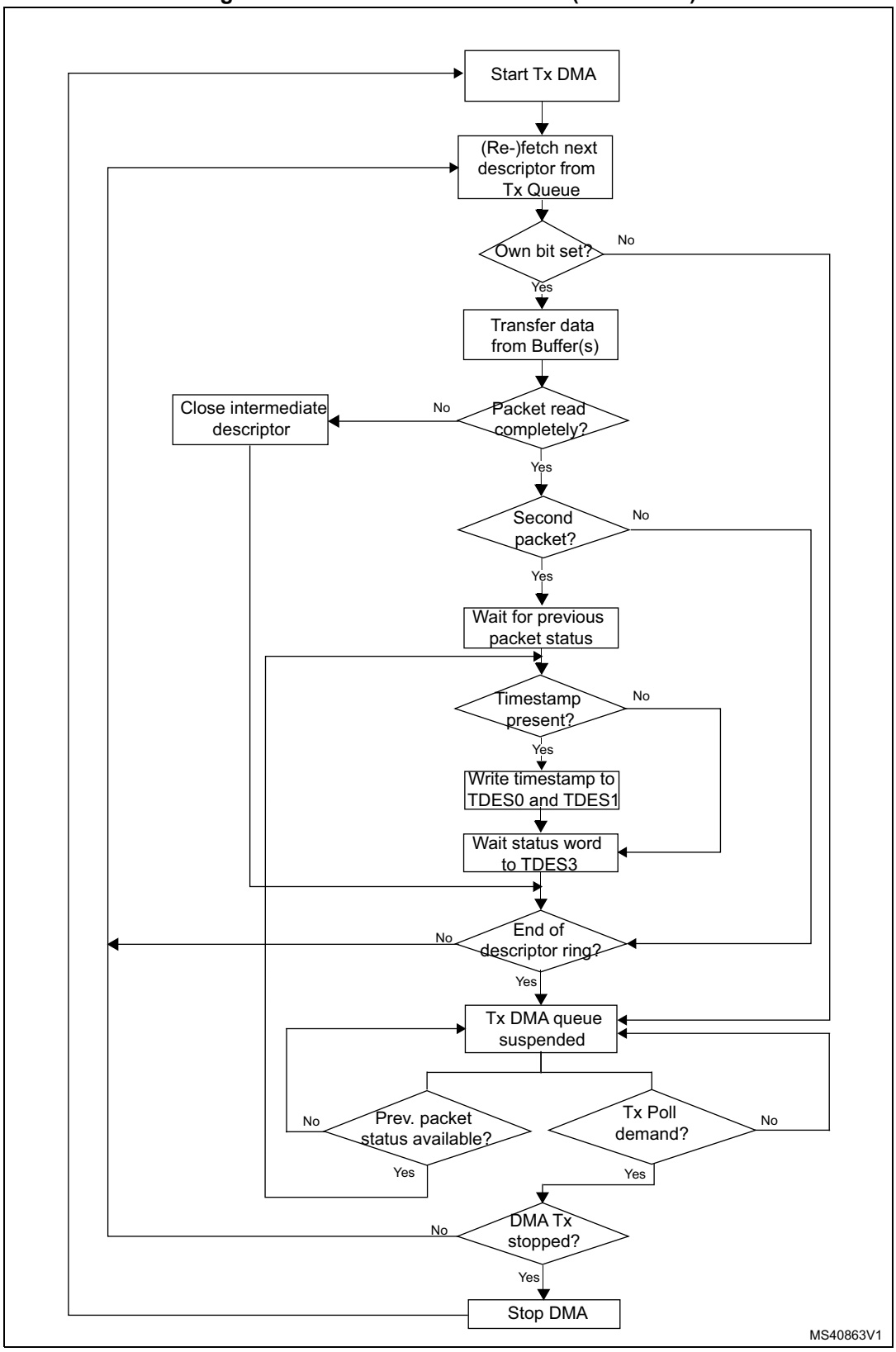
In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see *Section : DMA transmission in default mode*).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:
   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.
   a) The DMA writes the status to the corresponding TDES3.
   a) The DMA sets the relevant interrupts and returns to Suspend mode.

   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in *Figure 805: Receive DMA flow*.

Figure 804. DMA transmission flow (OSP mode)



DMA reception

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

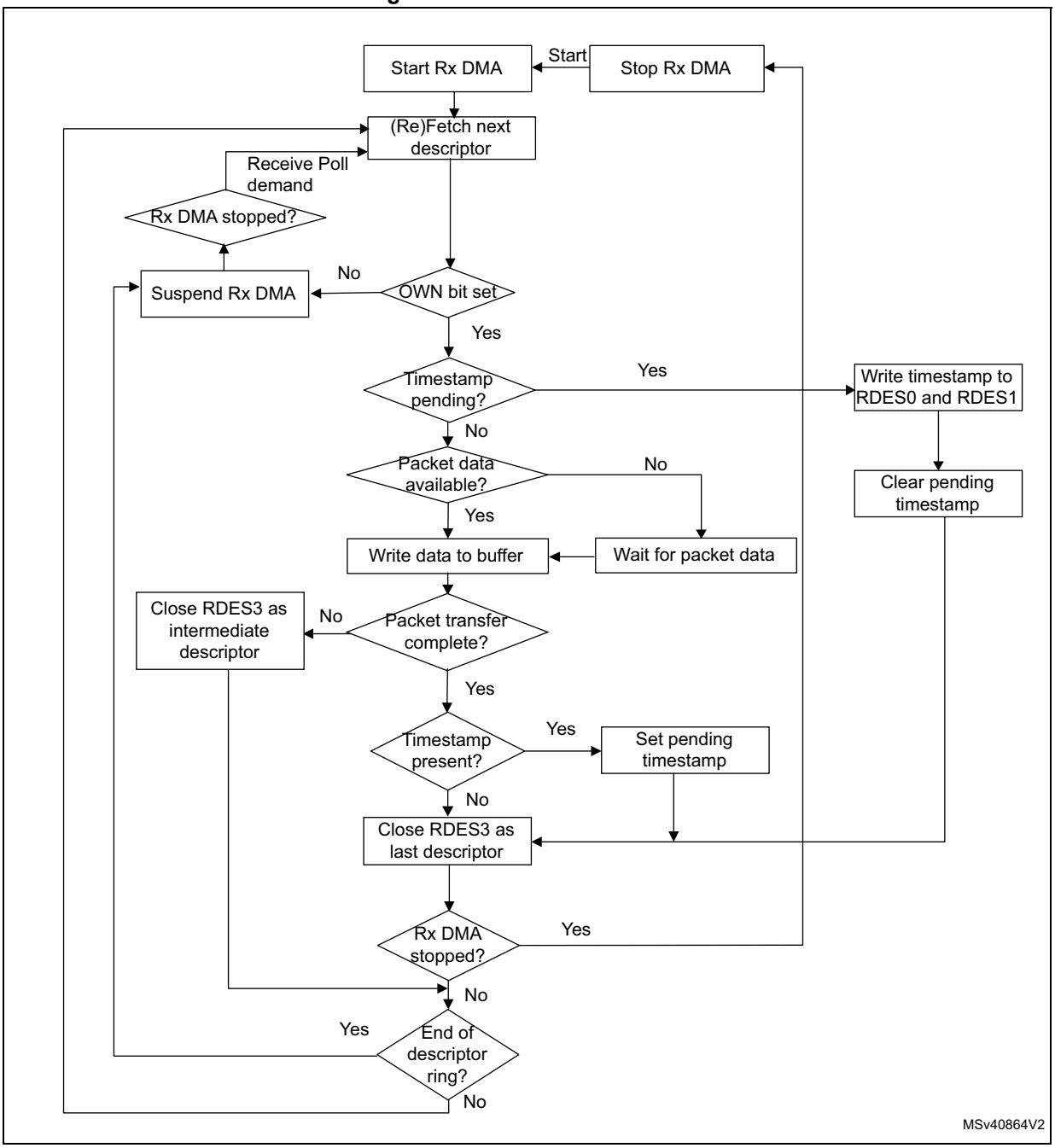
The DMA Rx descriptor ring structure is described in *Section 57.10: Descriptors*.

The reception sequence for Rx DMA engine is as follows (see also *Figure 805: Receive DMA flow*):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application should set the correct value in the Receive descriptor tail pointer register of corresponding DMA channel to indicate the location of the last valid descriptor for the DMA. If the tail pointer points to descriptor N, the last valid descriptor for the DMA is descriptor N - 1.
2. When bit 0 of *Channel receive control register (ETH_DMACRXCR)*) is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. The descriptors referenced between the current descriptor and the tail pointer registers are available for the DMA. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).
5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA retrieves the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.
8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. The DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).
11. The Receive DMA exits the Suspend state when a Receive Poll demand is given, and the application updates the channel Receive descriptor tail pointer register to indicate the location of the last valid descriptor for DMA. Then, the engine proceeds to step 2 and fetches again the next descriptor.

Note: Refer to Section : Descriptor tail pointer handling examples for updating the correct value in receive descriptor tail pointer register.

Figure 805. Receive DMA flow



Priority scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channel 0 to access descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the *DMA mode register (ETH_DMAMR)* specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a given channel.

If the Tx DMA and Rx DMA of a given channel are enabled, the DMA which gets the bus when the channel gets control of the bus must be specified. The priority between the

corresponding Tx DMA and Rx DMA can be configured through the TXPR field of the *DMA mode register (ETH_DMAMR)*. For round-robin arbitration, the weighted priority between the Tx DMA and Rx DMA is configured through the PR field of the *DMA mode register (ETH_DMAMR)*. *Table 639* provides information about the priority scheme between Tx DMA and Rx DMA.

Table 639. Priority scheme for Tx DMA and Rx DMA

DMA mode register (ETH_DMAMR)					Priority scheme
PR[2:0]			TXPR	DA	
x	x	x	0	1	Rx always has priority over Tx
0	0	0	0	0	Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.
0	0	1	0	0	Rx has priority over Tx in ratio 2:1.
0	1	0	0	0	Rx has priority over Tx in ratio 3:1.
0	1	1	0	0	Rx has priority over Tx in ratio 4:1.
1	0	0	0	0	Rx has priority over Tx in ratio 5:1.
1	0	1	0	0	Rx has priority over Tx in ratio 6:1.
1	1	0	0	0	Rx has priority over Tx in ratio 7:1.
1	1	1	0	0	Rx has priority over Tx in ratio 8:1.
x	x	x	1	1	Tx always has priority over Rx.
0	0	0	1	0	Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.
0	0	1	1	0	Tx has priority over Rx in ratio 2:1.
0	1	0	1	0	Tx has priority over Rx in ratio 3:1.
0	1	1	1	0	Tx has priority over Rx in ratio 4:1.
1	0	0	1	0	Tx has priority over Rx in ratio 5:1.
1	0	1	1	0	Tx has priority over Rx in ratio 6:1.
1	1	0	1	0	Tx has priority over Rx in ratio 7:1.
1	1	1	1	0	Tx has priority over Rx in ratio 8:1.

57.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 32-bit wide data paths: the Transmit path and the Receive Path.

- Transmit path

The application or internal DMA pushes the Ethernet packets read from the application or system memory into the Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 2048 bytes.

- Receive path

  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured Receive threshold (RTC bits[1:0] defined in *Rx queue operating mode register (ETH_MTLRXQOMR)*), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 2048 bytes.

57.4.3 MAC

The MAC is responsible of the Ethernet protocol processing.In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

MAC transmission

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   - The MAC completes the normal transmission and provides the transmission status to the MTL.
   - If a normal collision (in Half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:

     the packet was successfully transmitted;

     the maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) is observed from the MAC.
   - If any one of the following event happens, the MAC aborts the packet transmission:

     no carrier (Half-duplex mode)

     loss of carrier (Half-duplex mode)

     excessive deferral (Half-duplex mode)

     late collisions (Half-duplex mode)

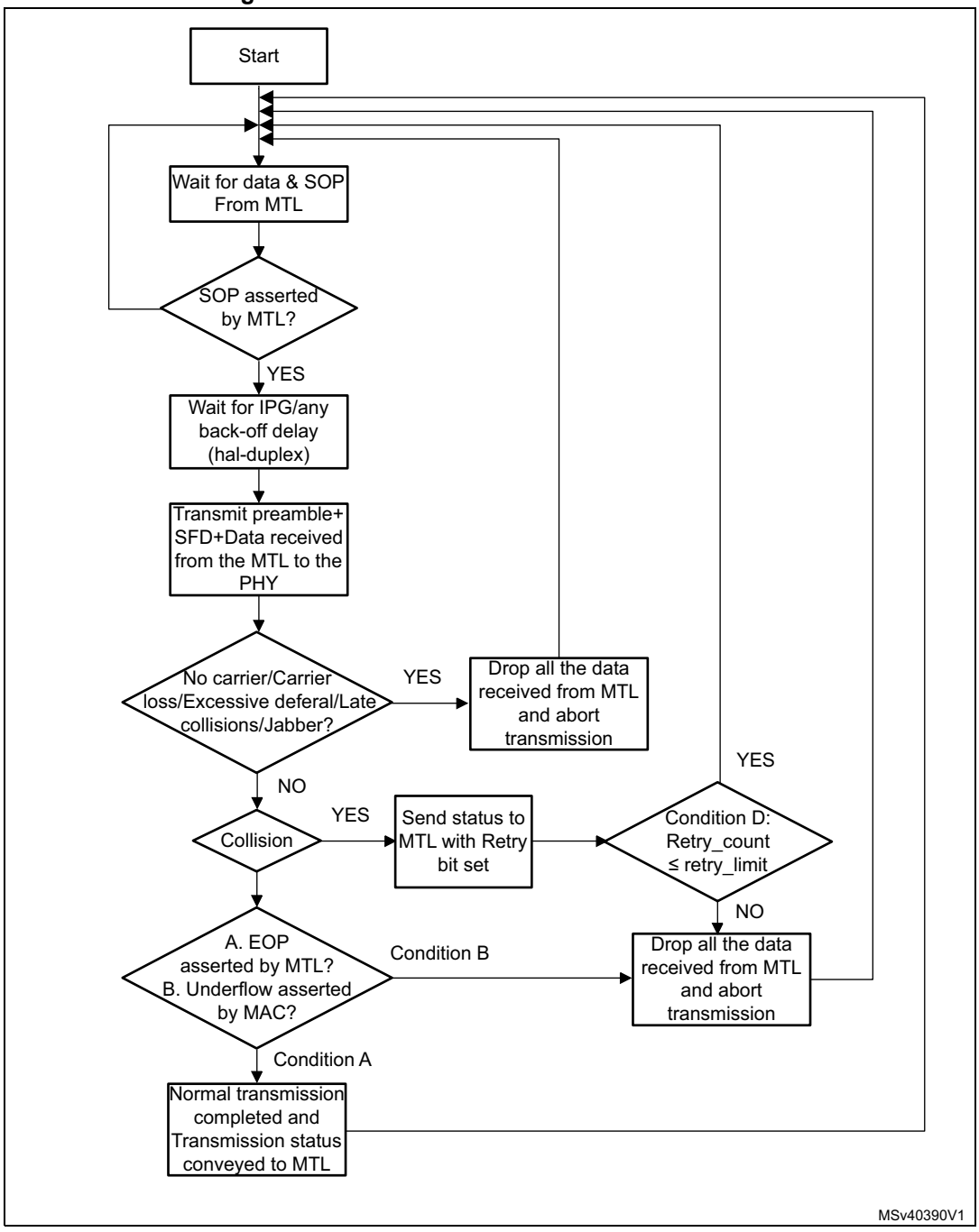
jabber

the MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.
5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

Figure 806: Overview of MAC transmission flow illustrates the MAC transmission process flow.

Figure 806. Overview of MAC transmission flow



MAC reception

A receive operation is initiated when the MAC detects an SFD on MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

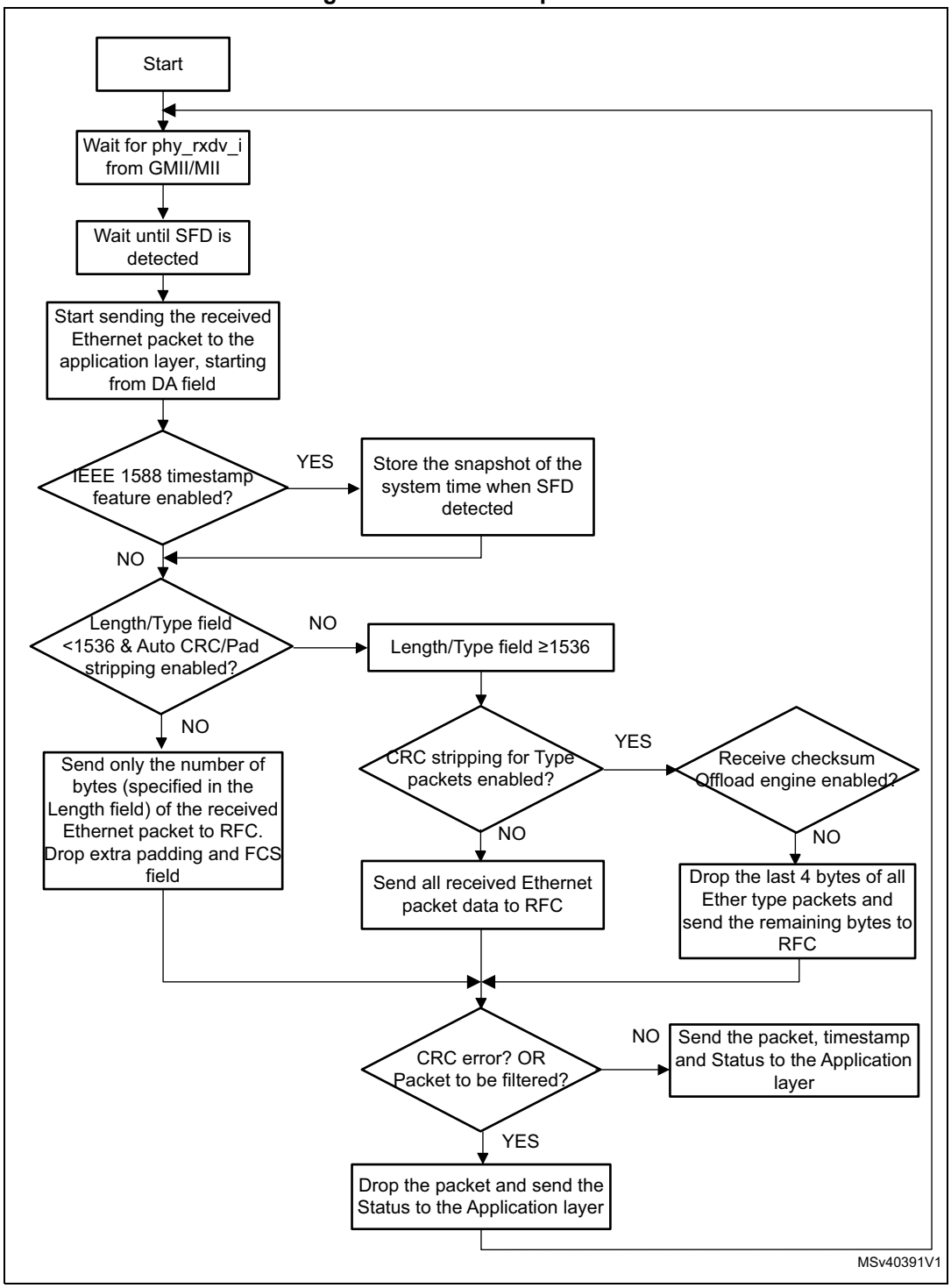
The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of MII becomes active, the Receive State Machine (RSM) starts looking for the SFD field (0xD nibble).

   The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).
3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application. The MAC converts the received nibble data into bytes and forwards the valid packet data to the RFC module.
4. The receive state machine decodes the Length/Type field of the Ethernet packet being received.

   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the *Operating mode configuration register (ETH_MACCR)*), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.
5. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the *Operating mode configuration register (ETH_MACCR)*. However, if the CRC stripping has been enabled for Type packets and not enabled the Receive Checksum Offload Engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.
6. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of *Watchdog timeout register (ETH_MACWTR)*) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of *Operating mode configuration register (ETH_MACCR)*. However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.

Figure 807. MAC reception flow



57.5 Ethernet functional description: MAC

57.5.1 Double VLAN processing

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

Transmit path

Table 640: Double VLAN processing features in Tx path describes the features supported by the MAC on the Transmit side.

Table 640. Double VLAN processing features in Tx path

Feature	Description
Support for C-VLAN and S-VLAN Tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of *VLAN inclusion register (ETH_MACVIR)* and *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN Tag deletion	VLAN tag deletion can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN Tag Insertion or Replacement	VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the control word.

Receive path

Table 641: Double VLAN processing in Rx path describes the features supported by the MAC on the Receive side and the corresponding bits in the *VLAN tag register (ETH_MACVTR)*.

Table 641. Double VLAN processing in Rx path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN Tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.
16-bit outer and inner VLAN Tag and Type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN Tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

57.5.2 Source address and VLAN insertion, replacement, or deletion

Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets

  To enable this feature for all packets, program the SARC field of the *Operating mode configuration register (ETH_MACCR)*.
- Enabling SA insertion or replacement for selective packets

  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of Transmit Descriptor Word 3/TDES3, refer to *Section 57.10.3: Transmit descriptor*) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or

replacement by MAC Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

VLAN insertion, replacement, or deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

  Insertion or replacement is performed based on the setting of VLTI bit in the *VLAN inclusion register (ETH_MACVIR)* as described in *Table 642: VLAN insertion or replacement based on VLTI bit*.

Table 642. VLAN insertion or replacement based on VLTI bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with VT field of Transmit context descriptor of the packet
VLTI bit is reset	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with the VLT field of *VLAN inclusion register (ETH_MACVIR)*

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of *VLAN inclusion register (ETH_MACVIR)*.
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see *Table 675: TDES2 normal descriptor (read format)*).

  In addition, the VLP (VLAN Priority control) bit must be reset in *VLAN inclusion register (ETH_MACVIR)* (for outer VLAN) and *Inner VLAN inclusion register (ETH_MACIVIR)* (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.

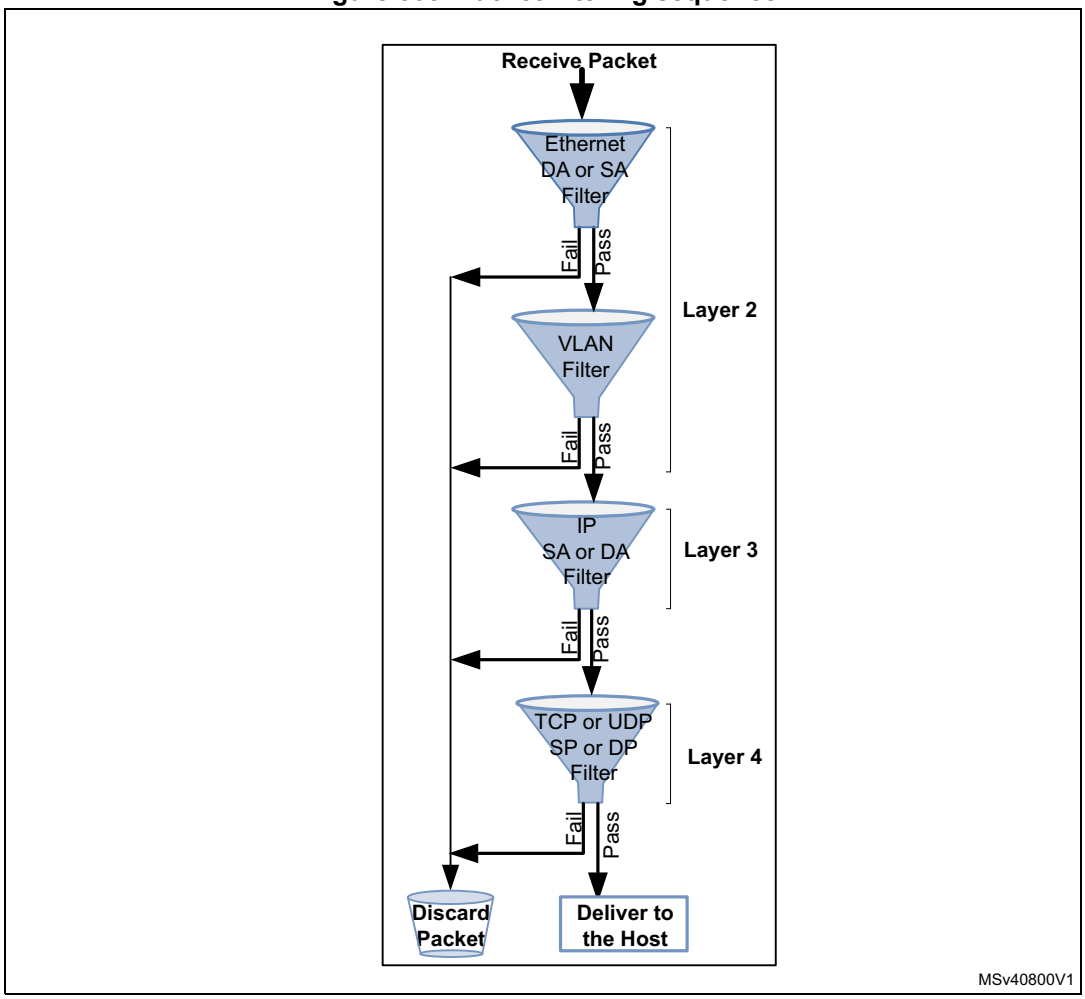
57.5.3 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- *MAC source or destination address filtering*: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- *VLAN filtering*: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- *Layer 3 and Layer 4 filtering*: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 808* shows the filtering sequence for Rx packets.

Figure 808. Packet filtering sequence



The sequence shown in *Figure 808* is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the Layer filters are not enabled, that filter is bypassed and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control.

For example, when RA bit of *Packet filtering control register (ETH_MACPFR)* is set to 1, all the discarded packets are forwarded to the host but with their packet status indicating the

specific filter failure. If RA bit is cleared to 0, VTFE and IPFE bits of *Packet filtering control register (ETH_MACPFR)* control if the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

MAC source or destination address filtering

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast destination address filtering

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of *Packet filtering control register (ETH_MACPFR)* is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in *MAC Address x high register (ETH_MACAxHR)*. This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

Multicast destination address filtering

To program the MAC to pass all multicast packets, set the PM bit in *Packet filtering control register (ETH_MACPFR)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the *Packet filtering control register (ETH_MACPFR)*.

In Perfect filtering mode, the multicast address is compared with the programmed MAC destination address registers. Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

Hash or Perfect address filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Packet filtering control register (ETH_MACPFR)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

Broadcast address filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in *Packet filtering control register (ETH_MACPFR)*.

Unicast source address filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of *MAC Address x high register (ETH_MACAxHR)*.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in *Packet filtering control register (ETH_MACPFR)*. Otherwise, the result of the SA filter is given as a status bit in the Receive Status word (see *Table 644*). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

Inverse filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of *Packet filtering control register (ETH_MACPFR)*. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 643 and *Table 644* summarize the DA and SA filtering based on the type of packets received.

Note: When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

Table 643. Destination address filtering

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 643. Destination address filtering (continued)

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Multicast	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 644. Source address filtering

Packet type	PR	SAIF	SAF	SA filter operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN filtering

The MAC supports Perfect and Hash VLAN filtering. Refer to *Section 57.9.15: Programming guidelines to perform VLAN filtering on the receiver* for detailed programming steps.

VLAN tag Perfect filtering

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of *VLAN tag register (ETH_MACVTR)*. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of *VLAN tag register (ETH_MACVTR)*. The VLAN packet status bit(bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

VLAN tag Hash filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN tag Hash filtering feature can be enabled using the VTHM (VLAN tag Hash Table match enable) bit of the *VLAN tag register (ETH_MACVTR)*. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag register (ETH_MACVTR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 645 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of *Packet filtering control register (ETH_MACPFR)* is set, all packets are received and the VLAN match status is indicated in the VF bit of *RDES2 normal descriptor (write-back format)*. When the RA bit is not set and the VTFE bit is set in *Packet filtering control register (ETH_MACPFR)*, the packet is dropped if the final VLAN match status is Fail. In *Table 645*, value *X* means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of *VLAN tag register (ETH_MACVTR)*, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in *VLAN tag register (ETH_MACVTR)*.

Table 645. VLAN match status(1)

VID	VLAN perfect filter match result	VTHM Bit	VLAN Hash filter match result	VTIM bit	Final VLAN match status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, ‘X’ represents any value.

Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the *Operating mode configuration register (ETH_MACCR)* to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets

  The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of *Operating mode configuration register (ETH_MACCR)* is set and one of the following conditions is true:

  – All enabled Layer 3 and Layer 4 fields match.

  – At least one of the enabled field matches and other fields are bypassed or disabled

    When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets

  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets.The aborted or partial packets are dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programmed to a small value. Such packet transfer to the application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

- Non-TCP or UDP IP Packets

  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

Layer 4 filtering

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 filters register set

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- *Layer4 Address filter 0 register (ETH_MACL4A0R)*
- *Layer3 Address 0 filter 0 register (ETH_MACL3A00R)*
- *Layer3 Address 1 filter 0 register (ETH_MACL3A10R)*
- *Layer3 Address 2 filter 0 register (ETH_MACL3A20R)*
- *Layer3 Address 3 filter 0 register (ETH_MACL3A30R)*

The second, and independent set of registers are: *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, *Layer 4 address filter 1 register (ETH_MACL4A1R)*, *Layer3 address 0 filter 1 Register (ETH_MACL3A01R)*, *Layer3 address 1 filter 1 register (ETH_MACL3A11R)*, *Layer3 address 2 filter 1 Register (ETH_MACL3A21R)* and *Layer3 address 3 filter 1 register (ETH_MACL3A31R)*.

57.5.4 IEEE 1588 timestamp support

The IEEE 1588 standard defines a precision time protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources.

This chapter contains the following sections:

- *IEEE 1588 timestamp support*
- *IEEE 1588 system time source*
- *IEEE 1588 auxiliary snapshots*
- *Flexible pulse-per-second output*

- *PTP timestamp offload function*
- *One-step timestamp*"

IEEE 1588 timestamp support

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588 2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement subsecond time in digital or binary format

Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- Ordinary clock

  The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

  The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

  – Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in *Timestamp control Register (ETH_MACTSCR)*.

  – Maintenance of the data sets such as timestamp values.

  The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 646. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

  For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in *Timestamp control Register (ETH_MACTSCR)*.

- Boundary clock

The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and signaling end in the protocol engine of the boundary clock. Such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

– The clock data sets are common to all ports of the boundary clock.

– The local clock is common to all ports of the boundary clock.

- End-to-end transparent clock

  The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the ingress port to the egress port.

  The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both ingress and egress ports only for the messages mentioned in *Table 647*. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the *Timestamp control Register (ETH_MACTSCR)*.

Table 647. End-to-end transparent clock: PTP messages for snapshot

PTP messages
SYNC
Delay_Req

- Peer-to-peer transparent clock

  In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

  The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

  The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in *Table 655*.

Table 648. Peer-to-peer transparent clock: PTP messages for snapshot

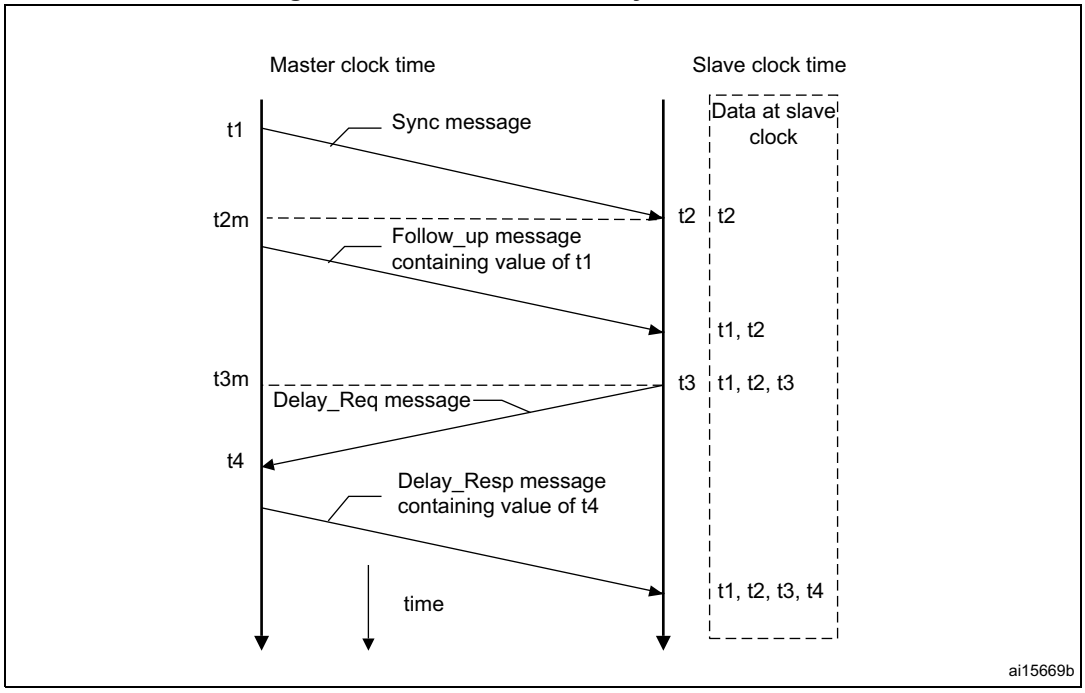
PTP messages
SYNC
Pdelay_Req
Pdelay_Resp

You can take the snapshot by setting the SNAPTYPESEL bit to 11 in *Timestamp control Register (ETH_MACTSCR)*.

Delay request-response mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 809* shows the process that PTP uses for synchronizing a slave node with a master node by exchanging PTP messages.

Figure 809. Networked time synchronization



As shown in *Figure 809*, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t_1. This time must be captured for Ethernet ports at MII.
2. The slave receives the SYNC message and also captures the exact time, t_2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, t_3, at which this packet leaves the MII interface.

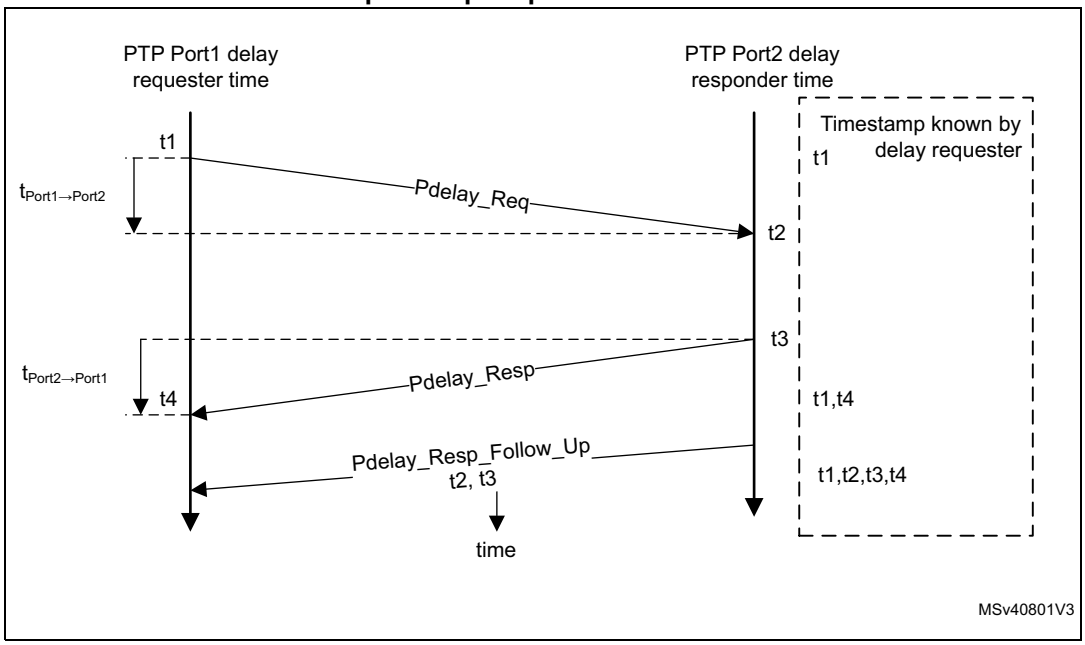
5. The master receives the message, capturing the exact time t_4, at which the message enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference with the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

Peer-to-peer PTP transparent clock (P2P TC) message support

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow_up, and Delay Response messages. *Figure 810* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 810. Propagation delay calculation in clocks supporting peer-to-peer path correction



As shown in *Figure 810*, the propagation delay is calculated as follows:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t_1) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t_2) for this message.

3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t_3) for this message.

   To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp message
   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp_Follow_Up message
   – Timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively
4. Port 1 generates a timestamp (t_4) on receiving the Pdelay_Resp message.
5. Port 1 uses all four timestamps to compute the mean link delay.

Timestamp correction

According to the IEEE 1588 specifications, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

As the MAC takes the timestamp at an internal point far from the actual boundary of the node and network, this captured timestamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (MII Tx, Rx clock) being different at the capture point as compared to the PTP clock (clk_ptp_ref_i) that is used to generate the time. The resultant CDC (clock domain crossing) circuits add an error the depends on the clock period of the MII and PTP clocks.

Ingress correction

In the Receive side, the timestamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet SFD bit is received at the port boundary. Therefore, the captured timestamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between boundary point and the input of the core

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from the PHY datasheet or timing characteristics.
2. Internal latency from the input of the core to the internal capture point

   The latency differs based on the active PHY interface (such as MII or RMII) and the operating speed, as shown in *Table 649*.
3. CDC synchronization

   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_ptp_ref_i).

The values determined from these three components should be added by the software and must be written into the TSIC field of the *Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)*.

Note: The value written to the register must be negative (two's complement), because it has to be subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set, the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1 ns. So bit 31 of TSIC must be set to 1 (for negative value) and bits 30 to 0 must be programmed with "10^9 – total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is –5 ns, then the value is 0xBB9A C9FB.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset, the nanoseconds field of the captured timestamp is in binary format with a granularity of ~0.466 ns. Therefore, bits[30:0] must be written with "2^{31} – total ingress_correction_value" represented in binary with bit[31] = 1.

Egress correction

In the Transmit side, the timestamp captured at the internal snapshot point is earlier (advanced in time) as compared to the time at which that packet SFD bit is output at the port boundary. Therefore, the captured timestamp must be compensated by the egress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp egress correction nanosecond register (ETH_MACTSECNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), the egress latency must be determined from the PHY datasheet or timing characteristics.
2. Internal latency from the internal capture point and the output of the core

   The latency differs based on the active PHY interface (RMII, MII, etc.) and the operating speed as shown in *Table 649*.
3. CDC synchronization error

   The timestamp correction because of synchronization is compensated by adding

$$\text{EGRESS_SYNC_CORR} = (1 \times \text{PTP_CLK_PER} + 4 \times \text{TX_CLK_PER})$$

Table 649 lists the egress and ingress latency values for various PHY interfaces:

Table 649. Egress and ingress latency for PHY interfaces

PHY interface		Egress latency	Ingress latency
RMII	100 Mbps	40	120
RMII	10 Mbps	400	800

Frequency range of reference timing clock

The timestamp information are transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of MII and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

The PTP clock frequency limitations are the following:

- Maximum PTP clock frequency

  The maximum PTP clock frequency is limited by the maximum resolution of the reference time (10 ns at 100 MHz). In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

- Minimum PTP clock frequency

  The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time with the MII clock domain. This relationship is given by the following equation:

  3 × PTP clock period + 4 × MII clock period ≤ Minimum gap between two SFDs

  The MII clock frequency is fixed by IEEE specifications. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in *Table 650*.

Table 650. Minimum PTP clock frequency example

Mode	Minimum gap between two SFDs	Minimum PTP frequency with internal timestamp
10 Mbps Full-duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
10 Mbps Half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz
100 Mbps full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
100 Mbps Half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz

PTP processing and control

Table 651 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008 specifications.

Table 651. Message format defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField(1)								1	32
logMessageInterval								1	33

1. The controlField is used in version 1. In version 2, the messageType field is used for detecting different message types.

Some fields of the Ethernet payload can be used to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP packets over IPv4
- PTP frames over IPv6
- PTP packets over Ethernet

PTP packets over IPv4

Table 652 provides information about the fields that are matched to control the snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 651*.

Table 652. Message format defined in IEEE 1588-2008

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x0800	IPv4 datagram
IP version and header length	14	0x45	IP version is IPv4
Layer 4 protocol	23	0x11	UDP
IP multicast address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132

Table 652. Message format defined in IEEE 1588-2008 (continued)

Matched field	Octet position	Matched value	Description
IP multicast address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP Primary multicast address: 224.0.1.129 PTP Pdelay multicast address: 224.0.0.107
UDP destination port	36, 37	0x013F, 0x0140	0x013F: PTP event messages(1) 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

PTP frames over IPv6

Table 653 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 651*.

Table 653. IPv6-UDP PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x86DD	IP datagram
IP version	14 (bits [7:4])	0x6	IP version is IPv6
Layer 4 protocol	20(1)	0x11	UDP
PTP multicast address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)

Table 653. IPv6-UDP PTP packet fields required for control and status (continued)

Matched field	Octet position	Matched value	Description
UDP destination port	56, 57a	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	94a	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP message type field (IEEE 1588 version 2)	62a (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The Extension header is not defined for PTP packets.

PTP packets over Ethernet

Table 654 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

Table 654. Ethernet PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC destination multicast address(1)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses(2): 01-1B-19-00-00-00 01-80-C2-00-00-0E(3)
MAC packet type	12, 13	0x88F7	PTP Ethernet packet
PTP control field (IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management

Table 654. Ethernet PTP packet fields required for control and status (continued)

Matched field	Octet position	Matched value	Description
PTP message type field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the TSENMACADDR bit of *Timestamp control Register (ETH_MACTSCR)* is set.
2. IEEE 1588-2008, Annex F
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

Transmit path functions

The MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the MII interface. The packets, for which a timestamp has to be captured, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The packets for which a timestamp has to be captured must be specified. The packets can be defined by using the control bits in the Transmit Descriptor (see *Section 57.10.3: Transmit descriptor*). The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus automatically connecting the timestamp to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. The snapshot of the time to be sent to the application can be controlled by using the following options of the *Timestamp control Register (ETH_MACTSCR)*:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)

- Enable the node to be a master or slave and select the snapshot type
  This feature controls the type of messages for which snapshots are taken.

Note: The peripheral also supports the PTP messages over VLAN packets.

Table 655 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPSEL field in *Timestamp control Register (ETH_MACTSCR)*.

Table 655. Timestamp snapshot dependency on ETH_MACTSCR bits

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP messages
00	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	X	X	SYNC, Delay_Req
11	X	X	Pdelay_Req, Pdelay_Resp

The DMA returns the timestamp to the software application inside the corresponding Receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Programming guidelines for IEEE 1588 timestamping (system time correction)

See *Section : System time correction* in *Section 57.9.9: Programming guidelines for IEEE 1588 timestamping on page 2792*.

IEEE 1588 system time source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

Description of IEEE 1588 system time source

The peripheral uses the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:

- UInteger48 seconds field
  The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bit wide. For example, 2.000000001 seconds are represented as seconds Field = 0x0000 0000 0002.
- UInteger32 nanosecondsField
  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 seconds are represented as nanoSeconds = 0x0000 0001.

The nanoseconds field supports the following two modes:

– **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A C9FF, that is, (10e9-1) nanoseconds.

– **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF FFFF. Accuracy is ~0.466 ns per bit.

These modes can be set through TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)*.

System time register module

The 64-bit PTP time is updated using the PTP input reference clock, clk_ptp_ref_i. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being transmitted or received at the MII.

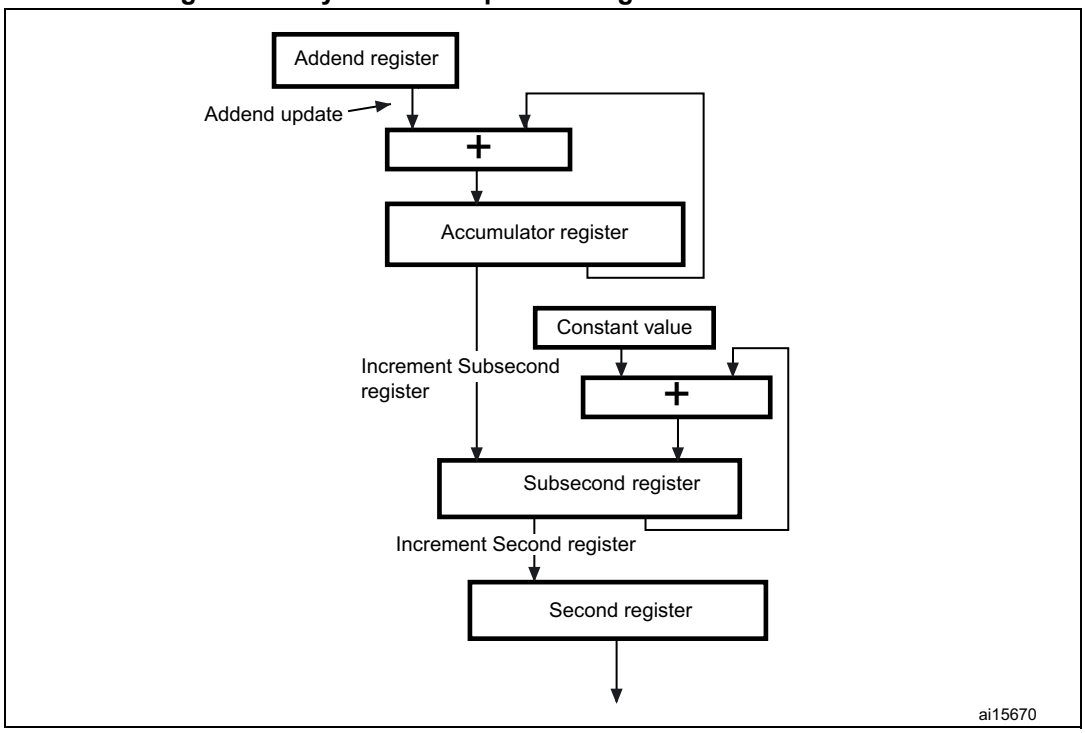
The system time counter can be initialized or corrected using either the coarse or the fine correction method.

In the coarse correction method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is programmed with the value in the timestamp update registers, whereas for system time correction the offset value (timestamp update register) is added to or subtracted from the system time.

In the fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588-2002 specifications) is corrected over a period of time, unlike in the coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in *Figure 811*. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in *Figure 811*.

Figure 811. System time update using fine correction method



The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC 4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B 7672. When there is not clock drift, the default addend value of 0xC1F0 7C1F (2^{32} / 1.32) must be programmed.

In *Figure 811*, the constant value used to accumulate the subsecond register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

1. At time $MasterSyncTime_n$ the master sends the slave clock a SYNC message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as follows:

$$MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$$

2. The master clock counts for current Sync cycle, $MasterClockCount_n$ is

$$MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$$

(assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

3. The slave clock count for current Sync cycle, $SlaveClockCount_n$ is

$$SlaveClockCount_n = SlaveClockTime_n - SlaveClockTime_{n-1}$$

4. The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is

$$ClockDiffCount_n = MasterClockTime_n - SlaveClockTime_n$$

5. The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is

$$FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$$

6. The frequency compensation value for Addend register, $FreqCompensationValue_n$ is

$$FreqCompensationValue_n = FreqScaleFactor_n \times FreqCompensationValue_{n-1}$$

In theory, this algorithm achieves the lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value by the master, the algorithm corrects it at the cost of additional Sync cycles.

Refer to *Section 57.9.9: Programming guidelines for IEEE 1588 timestamping* for detailed programming steps.

IEEE 1588 auxiliary snapshots

The auxiliary snapshot feature enables to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the eth_ptp_trgx (where x = 1 to 4) sideband signal.

Up to four auxiliary snapshot inputs can be configured and up to four snapshots can be stored. A FIFO is accessible through registers: *Auxiliary timestamp seconds register (ETH_MACATSSR)* and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*.

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the *Timestamp status register (ETH_MACTSSR)* to know the timestamp of which input is available for reading at the top of this FIFO.

When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the *Timestamp status register (ETH_MACTSSR)*. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in *Auxiliary control register (ETH_MACACR)*. When multiple snapshots are present in the FIFO, the count is indicated in bits[27:25] of *Timestamp status register (ETH_MACTSSR)*.

Flexible pulse-per-second output

The MAC supports either a fixed pulse-per-second output mode (also called fixed mode) or a flexible pulse-per-second output mode for the ETH_PPS_OUT and eth_ptp_pps_out outputs:

- Fixed pulse-per-second output

  In this mode, only the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the *PPS control register (ETH_MACPPSCR)*.
- Flexible pulse-per-second output

  In this mode, the software has the flexibility to program the start or stop time, width, and interval of the pulse generated on the eth_ptp_pps_out output:

  The start and stop times are programmed through *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*.

  The PPS width and interval are programmed in terms of granularity of system time (number of the units of subsecond increment value) through *PPS width register (ETH_MACPPSWR)* and *PPS interval register (ETH_MACPPSIR)*, respectively.

Note: By default, the peripheral is in Fixed mode and indicates one second interval. When Fixed mode is selected by clearing PPSEN0 to 0 in the PPS control register (ETH_MACPPSCR):

- *the output on all PPS outputs is controlled by the value programmed in the PPSCTRL_PPSCMD field. Independent control of individual PPS output is not supported in Fixed mode.*
- *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR) are used only for generating target time reached interrupt; they are not used for PPS output generation.*
- *TRGTMODSEL0/1/2/3 must be programmed to 0.*
- *the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).*

Description of flexible pulse-per-second (PPS) output

The peripheral supports the following features with the flexible PPS outputs:

- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, the stop time can be programmed before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of subsecond increment value programmed in the *Subsecond increment register (ETH_MACSSIR)*. The pulse width can be programmed from 1 to 232–1 units of subsecond increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment value. You can program the interval between pulses from 1 to 232–1 units of subsecond increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

Note: The PTP reference clock mentioned in the following sections is the clock at which the system time is updated. When the TSCFUPDT bit of Timestamp control Register (ETH_MACTSCR) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In Fine correction mode, this is the clock tick at which the system time is updated (using Subsecond increment register (ETH_MACSSIR) (as shown in Figure 811).

Refer to *Section 57.9.12: Programming guidelines for flexible pulse-per-second (PPS) output* for further details on how configuring flexible pulse output.

PPS start and stop times

The initial start time can be programmed in the Target Time registers.

If required, the start or stop time can be programmed again. However, this can be done only after the earlier programmed value is synchronized with the PTP clock domain. Bit 31 of *PPS target time nanoseconds register (ETH_MACPPSTTNR)* indicates that the synchronization is complete. This enables to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS width and interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to obtain a PPS pulse width of 40 ns and an interval of 100 ns with a PTP reference clock of 50 MHz, program the width and interval to values 2 and 5, respectively. Smaller granularity can be achieved by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, program or update the interval and width of the PPS signal output.

PTP timestamp offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network.

These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

The PTP offload feature is selected through *PTP Offload control register (ETH_MACPOCR)*. 80-bit PTP node identity is configured through the following three registers: *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.

Description of PTP offload function

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. *Table 656* indicates the PTP message generation criteria.

Table 656. PTP message generation criteria

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
00	0	1	Ordinary or Boundary Slave	SYNC message reception	Delay_Req
00	1	1	Ordinary or Boundary Master	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
01	0	1	Transparent Slave	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req

Table 656. PTP message generation criteria (continued)

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
01	1	1	Transparent Master	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
11	X	X	Peer-to-Peer Transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
All other programming combinations are invalid for PTP Offload feature.					

Note: Clocks supporting peer delay mechanism must not generate delay request/delay response messages, according to IEEE 1588-2008 specifications. However, the peripheral controller supports this for flexibility, with a programmable control bit (DRRDIS) in the PTP Offload control register (ETH_MACPOCR).

The DRRDIS bit can be used to control the response generation for delay request/delay response message. For example, in transparent slave mode, delay request is generated in response to received SYNC *only when the bit is reset.*

When the MAC is set as an Ordinary or Boundary Slave clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission of the corresponding Delay_Req message. Similarly, various other modes of operation are explained in *Table 656*.

The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the Destination Address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B 1900 0000 for all except peer delay mechanism messages and 0x0180 C200 000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with unicast destination address, it takes the SA field of the received packets and makes

them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all the received PTP messages are forwarded to the application along with Rx status, indicating whether the response was generated by the MAC, if it satisfies the packet filtering logic of the MAC receiver

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress timestamp of these two PTP messages are provided in the Tx TS status (Tx Timestamp Status register and interrupt generated).

In addition to messageType and versionPTP fields match for basic PTP over Ethernet message detection, the following additional fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one-step indication (0b0).
3. The transportSpecific field is checked for Default PTP over Ethernet (0b0000) or 802.1AS mode (0b1111) when enabled.

PTP packet generation

This section explains the format and content of the automatically generated PTP packets by the MAC when this mode is enabled. It provides the template of the common PTP message header, as well as the detailed description of the fields of the specific PTP packets generated.

Table 657. Common PTP message header fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

PTP message header fields

- **messageType**

  The following encoded values are used for PTP message types:
  - SYNC: 0000
  - Delay_Req: 0001
  - Pdelay_Req: 0010
  - Pdelay_Resp: 0011
  - Delay_Resp: 1001
- **transportSpecific**

  The following transport protocol encoding is used:
  - Default PTP over Ethernet: 0000
  - 802.1AS mode: 0001
- **versionPTP**

  It is always set to 2 because PTP version 2 is supported.
- **domainNumber**

  This field contains the value from the *PTP Offload control register (ETH_MACPOCR)*.
- **flagField**

  The following values are used:
  - alternateMasterFlag (Octet 0 bit 0): 0 for SYNC and Delay_Resp
  - twoStepFlag (Octet 0 bit 1): 0 for SYNC and Pdelay_Resp
  - unicastFlag (Octet 0 bit 2): 0 for Multicast Address, 1 for Unicast Address
- **correctionField**

  For more information, see *Table 658*.
- **sourcePortIdentity**

  This field takes the value programmed in the *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.
- **sequenceId**

  Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.
- **controlField**

  The following encoded values are used for controlField:
  - SYNC: 0000 0000
  - Delay_Req: 0000 0001
  - Pdelay_Req: 0000 0010
  - Pdelay_Resp: 0000 0101
  - Delay_Resp: 0000 0011
- **logMessageInterval**
  - SYNC:

    This field contains logSyncInterval from the corresponding MAC_Log_Message_Interval register.

– Delay_Resp:

This field contains the sum of DRSYNCR and logSyncInterval value taken from the *Log message interval register (ETH_MACLMIR)* for a multicast PTP message and 0111 1111 for unicast PTP message.

– Delay_Req, Pdelay_Req and Pdelay_Resp: 0111 1111

where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of –15 to 15 for logSyncInterval fields, which translates to a range from 32.768 micro second (2–15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- $2^{(30+N)}$ ns, when N is negative (–1 to –15)
- 2^{N} seconds, when N is positive (0 to 15)

For example:

- When logSyncInterval is programmed to 1, the interval is 2^{1}; therefore, the SYNC message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is 2^{-1} = 0.536 seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because 2^{-30} = 1 ns.
- When logSyncInterval is programmed to –-5, the interval is 2^{-5} = 33.55 ms; therefore, the SYNC message is sent once every 33.55 ms.

Note: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation $2^{(30+N)}$, because of the non-binary nature of the nanoseconds field of the system time.

PTP message-specific fields

The message-specific fields are the following:

- **messageLength**

  There is no suffix supported, so this field contains the length of the PTP message that includes 34-byte PTP common header and the body specific to the message type.

  For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.

- **originTimestamp**

  This field is the captured egress timestamp for SYNC, Delay_Req, and Pdelay_Req PTP messages.

- **receiveTimestamp**

  For Delay_Resp PTP message, this is the ingress timestamp of the corresponding received Delay_Req PTP message.

- **requestingPortIdentity**

  For Delay_Resp and Pdelay_Resp PTP messages, this is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.

- **requestReceiptTimestamp**

  For the Pdelay_Resp PTP message, this field is set to 0.

One-step timestamp

The MAC supports the one-step timestamp feature that enables to identify the offset in the packet and inserts the timestamp received from the application at that offset.

MAC Transmit PTP mode for one-step timestamp

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 658 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the *Timestamp control Register (ETH_MACTSCR)* and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 658. MAC Transmit PTP mode and one-step timestamping operation

Programming			Mode	Per packet control(1)			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE (2)	OSTC (3)	TTS(4)	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)
00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
							Delay_Req (correction field for residence time and egress asymmetric correction)
00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
01	X	0	End-to-end Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Pdelay_Req (correction field for residence time and egress asymmetric correction)
						Ingress TS	Pdelay_Resp (correction field for residence time and ingress asymmetric correction)

Table 658. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[1]			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
01	0	1	Ordinary or Boundary Slave with support for peer delay mechanism or Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction) (applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
10	X	X	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Delay_Req (correction field for residence time and egress asymmetric correction)

Table 658. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control(1)			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE (2)	OSTC (3)	TTS(4)	
11	X	X	Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)

1. The per-packet control values provided here are the recommended settings used by devices in typical PTP operation and for the programmed mode.
2. TTSE represents TTSE bit of TDES2 transmit normal descriptor. The TTSE function is independent from the OST function and the programmed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of TDES3 transmit context descriptor
4. TTS represents the timestamp value provided in the TTSH, TTSL fields of TDES0 and TDES1 transmit normal descriptor (write-back format).

Note: Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp. Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

Enabling one-step timestamp

The one-step timestamp feature can be enabled for a given packet by setting bit 20 (OSTC) in TDES3 context descriptor. To update the correction field in certain PTP packets, the ingress timestamp must be given in the TSSL and TSSH fields.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

57.5.5 Checksum offload engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the Receive path.

Transmit checksum offload engine

In the transmit path, the MAC calculates the checksum and inserts it in the Tx packet. This feature helps reducing the load on the software and can improve the overall system throughput.

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, the Tx FIFO automatically operates in the Store-and-forward mode even if the MAC is configured in Threshold (cut-through) mode.

Make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter, the reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the checksum insertion must be enabled only in the packets that are less than the number of bytes, given by the following equation:

$$\text{Packet size} < \text{TxQSize} - (\text{PBL} + 7) \times 4$$

where

TxQSize corresponds to the TQS bitfield of *Tx queue operating mode register (ETH_MTLTXQOMR)*

PBL corresponds to the TxPBL bitfield of *Channel transmit control register (ETH_DMACTXCR)*

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

Note: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status it in the Transmit status (bit 0 in *Table 680: TDES3 normal descriptor (write-back format)*).

This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as

indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in *Table 680: TDES3 normal descriptor (write-back format)*). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 659 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: Do not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Section : Transmit checksum offload engine because it might result in incorrect checksum insertion or unexpected behavior.

Table 659. Transmit checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Authentication – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – No – No – Yes – No – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive checksum offload engine

The Receive Checksum Offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

The Receive Checksum Offload Engine (Rx COE) can be enabled by setting the IPC bit of *Operating mode configuration register (ETH_MACCR)*. When this bit is set, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the *VLAN tag register (ETH_MACVTR)* is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when DIS_TCP_EF bit of the *Rx queue operating mode register (ETH_MTLRXQOMR)* is reset and FEP bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 660: Receive checksum offload engine functions for different packet types describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 660. Receive checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – Yes – No – Yes – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

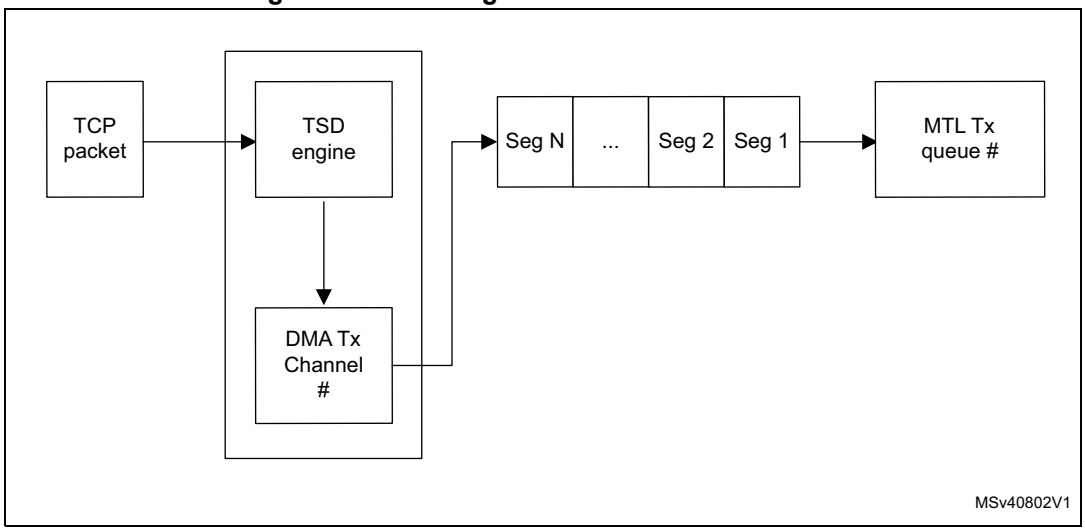
57.5.6 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in *Figure 812*.

This feature is enabled by programming the TSE bit of corresponding ETH_DMACCR register (see *Channel transmit control register (ETH_DMACTXCR)*). It is only supported when the MAC operates in Full-duplex mode.

For detailed programming steps, refer to *Section 57.9.14: Programming guidelines for TSO*.

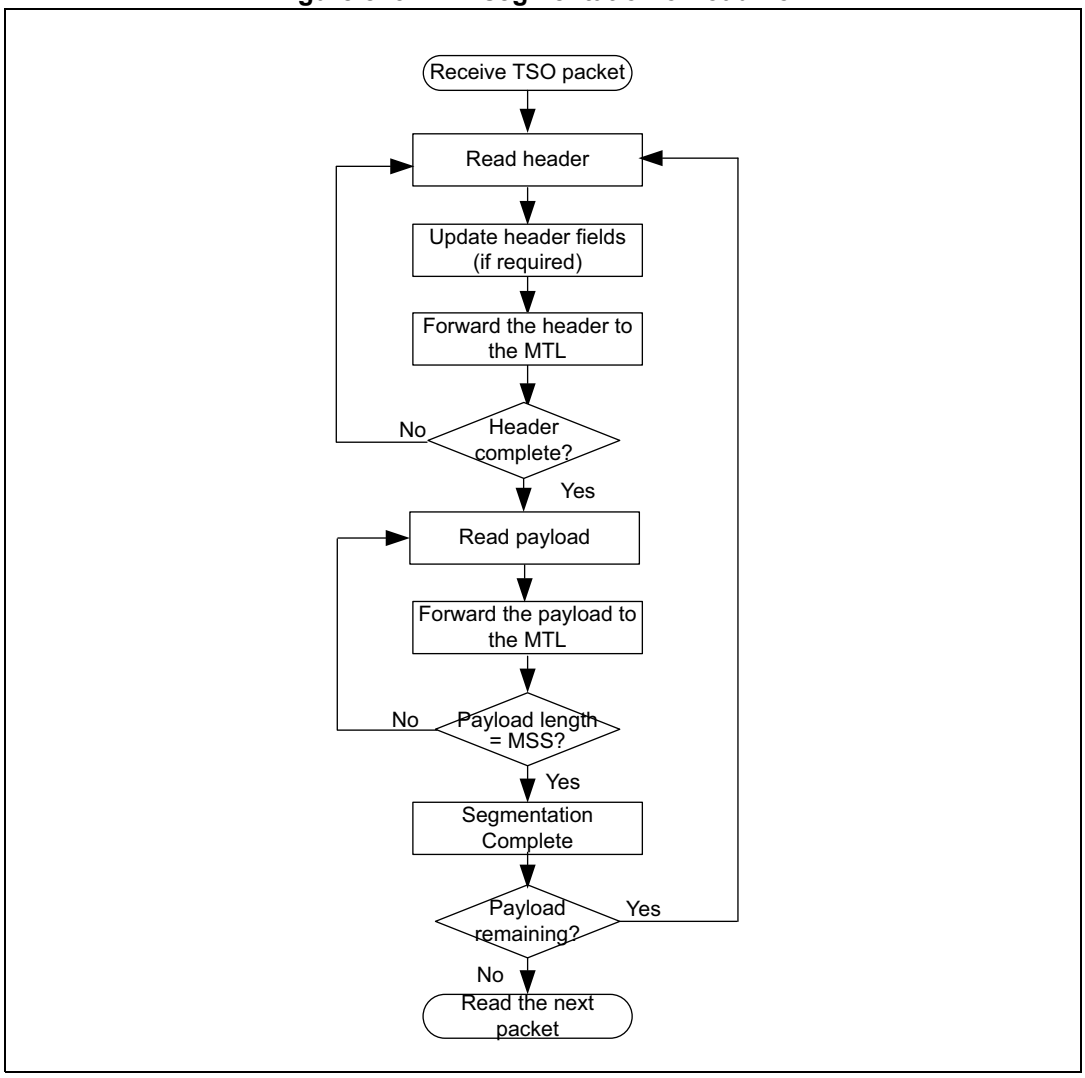
Figure 812. TCP segmentation offload overview



DMA operation with TSO feature

Figure 813 shows the TSO flow.

Figure 813. TCP segmentation offload flow



For the TSO feature, the Tx DMA operation is as follows:

1. The application sets up the Transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA Tx channel.
3. While in the Run state, the DMA fetches the next available descriptor and performs one of the following actions:
   – If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.

- If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a Context Descriptor Error (TDES3[23]) and fetches the next descriptor.
- If the descriptor is a normal descriptor, the DMA checks the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.
5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.
6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (if present and when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, it fetches again the header buffer in system memory, as done for the first segment. In such cases, the DMA does not close the first descriptor containing the header buffer until the header for last segment is fetched.
7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL Tx queue.
8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory, and directly pushes it into the MTL Tx queue. If the buffer(s) in the descriptor do(es) not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.
9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the Tx queue.
10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory), and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no more descriptor are available, the DMA enters the suspend state.

Note: The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first Normal Tx descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.

TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. *Table 661* describes how the TCP and IP headers are updated.

Table 661. TSO: TCP and IP header fields

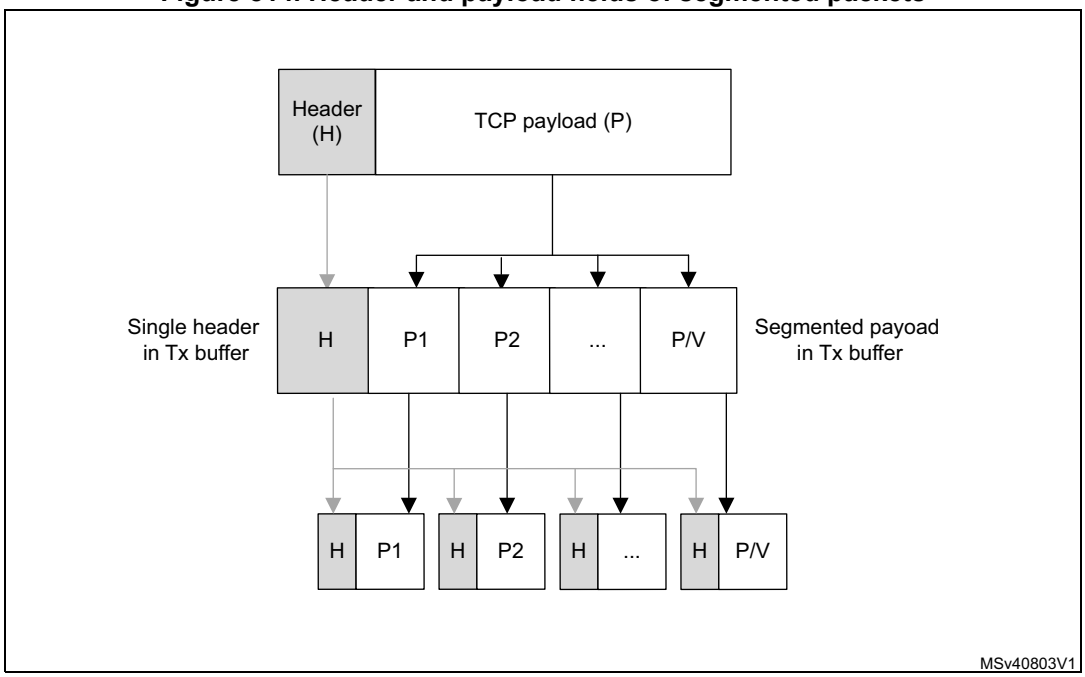
Packet sequence	TCP header	IP header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field is not modified. It is sent as per the header provided by the software. – IPv4 Header Checksum is recalculated. IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field = Previous Identification Field + 1 – IPv4 Header Checksum is recalculated IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = Remaining Payload + TCP Header Length + IP Header Length – Identification Field = Previous Identification Field + 1 – IPv4 header Checksum is recalculated IPv6 Header – Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

Header and payload fields of segmented packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in *Table 661: TSO: TCP and IP header fields*. *Figure 814: Header and payload fields of segmented packets* shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

Figure 814. Header and payload fields of segmented packets



Context descriptor sequence

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to *Section 57.10.3: Transmit descriptor*).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see *Channel interrupt enable register (ETH_DMACIER)*).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see *Channel status register (ETH_DMACSR)*).

Building the Descriptor and the packet for the TSO feature

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see *Section 57.10.3: Transmit descriptor*). If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMACCR (see *Channel control register (ETH_DMACCR)*) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMACTXCR register (see *Channel transmit control register (ETH_DMACTXCR)*) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQOMR register (see *Tx queue operating mode register (ETH_MTLTXQOMR)*). A MSS plus header equal to half the programmed Tx queue size is recommended.

The DMA also supports segmentation of VLAN-tagged TCP/IP frames. If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning this is an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 00 (MAC does CRC and padding) and checksum insertion control bits are forced to 11 (hardware does the checksum calculation for both header and payload).

57.5.7 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets**.** ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:

1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to the application with ARP Reply Not Generated status bit set (bit 34). However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request. If the Disable CRC check (DCRCC) bit of the *Extended operating mode configuration register (ETH_MACECR)* is set, then the MAC does not check for valid CRC of a ARP

request packet. It can generate an ARP response packet if the other conditions are valid. The ARP request packet must always have a valid CRC.

Note: When the received ARP request is less than the 64-byte packet length, the MAC does not send an ARP response. It is treated as a normal packet and forwarded to the application based on the MAC filter settings.

57.5.8 Loopback

The MAC supports loopback of transmitted packets to its receiver.

Guidelines for using loopback mode

Below some guidelines for using the loopback mode:

- Enable loopback only with the Full-duplex mode. In Half-duplex mode, the carrier sense signal or collision signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip, externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets since they may get corrupted in the loopback FIFO.

The Transmit and Receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the transmitted data to the Receive path. The FIFO is free-running to write on the write clock (eth_tx_clk) and read on every read clock (eth_rx_clk). At the start of each packet read from the FIFO, the write and read pointers are reinitialized to have an offset of four (in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer, and ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between MII Transmit and Receive clock frequencies.

Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module. The MAC does not process ARP or PMT packets that are looped back.

Enabling loopback mode

To enable this feature, program the LM bit of the *Operating mode configuration register (ETH_MACCR)*. Loopback can be enabled for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal Receive MII interface, irrespective of which PHY interface is selected.

The loopback data is also passed through the corresponding transmit PHY interface block, onto the Ethernet line.

Note: During loopback, the data/packet is reflected on mii_txd signal.

57.5.9 Flow control

The transmit flow control involves transmitting Pause packets in Full-duplex mode and backpressure in Half-duplex mode to control the flow of packets from the remote end. This section describes the flow control for transmit and receive paths.

Flow control in Full-duplex mode

In Full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. *Table 662* describes the fields of a Pause packet.

Table 662. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets
PT	Contains Pause time specified in the PT field of the *Tx Queue flow control register (ETH_MACQTXFCR)*

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Flow control in Half-duplex mode

In Half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 663 describes the flow control in the Tx path based on the setting of the following bits:

- TFE bit of *Tx Queue flow control register (ETH_MACQTXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Flow control is similar for all queues.

Table 663. Tx MAC flow control

TFE	DM	Description
0	X	The MAC transmitter does not perform the flow control or backpressure operation.
1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set.
1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue flow control register (ETH_MACQTXFCR)* is set.

Transmit flow control

The transmit flow control is enabled when TFE bit is set in *Tx Queue flow control register (ETH_MACQTXFCR)*.

Flow control trigger

The application can request the MAC to send a Pause packet or initiate back-pressure by setting the FCB bit in the corresponding *Tx Queue flow control register (ETH_MACQTX-FCR)*.

Receive flow control

In the Receive path, the flow control is functional only in Full-duplex mode. If any Pause packet is received in Half-duplex mode, the packet is considered as a normal control packet.

Description of receive flow control

Table 664 describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of *Rx flow control register (ETH_MACRXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Table 664. Rx MAC flow control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

The following sequence describes the Rx flow control:

1. The MAC checks the destination address (DA) of the received Pause packet for either of the following:
   - Multicast destination address: the DA matches the unique multicast address specified for the control packet (0x0180 C200 0001).
   - Unicast destination address: the DA matches the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x*

low register (ETH_MACAxLR)) and the UP bit of *Rx flow control register (ETH_MACRXFCR)* is set.

If the UP bit is set, the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.

2. The MAC decodes the following fields of the received packet:
   - Type field: this field is checked for 0x8808.
   - Opcode field: this field is checked for 0x0001 (Pause packet).
   - Pause time: the Pause time (for Pause packet) is captured to determine the time for which the transmitter needs to be blocked.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value.

Enabling receive flow control

Set the RFE bit in the *Rx flow control register (ETH_MACRXFCR)* to enable the Pause flow control.

57.5.10 MAC management counters

The peripheral supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC management counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in *Section 57.11.4: Ethernet MAC and MMC registers*.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set bit 0 in the *Packet filtering control register (ETH_MACPFR)*. The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

In addition to control registers, two sets of registers are implemented:

- Six registers used for collision, error, and good packet counters:
  - *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)*
  - *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)*
  - *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)*
  - *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)*
  - *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)*
  - *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)*
- Four registers to record LPI mode transition:
  - *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)*
  - *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)*
  - *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)*
  - *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)*

Definitions

The following terminology is used in MMC register descriptions:

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber timeout
  - No carrier or loss of carrier
  - Late collision
  - Packet underflow
  - Excessive deferral
  - Excessive collision
- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of range (non-Type packet only, longer than 1518 bytes)
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN frame maxsize = 1,522
  - Jumbo frame maxsize = 9,018
  - JumboVLAN frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the *Table 665*.

Table 665. Size of the maximum receive packet

JE	S2KP	GPSLCE	EDVLP	Untagged frame maximum size in bytes	Single VLAN frame maximum size in bytes	Double VLAN Frame maximum size in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

57.5.11 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the *Interrupt enable register (ETH_MACIER)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

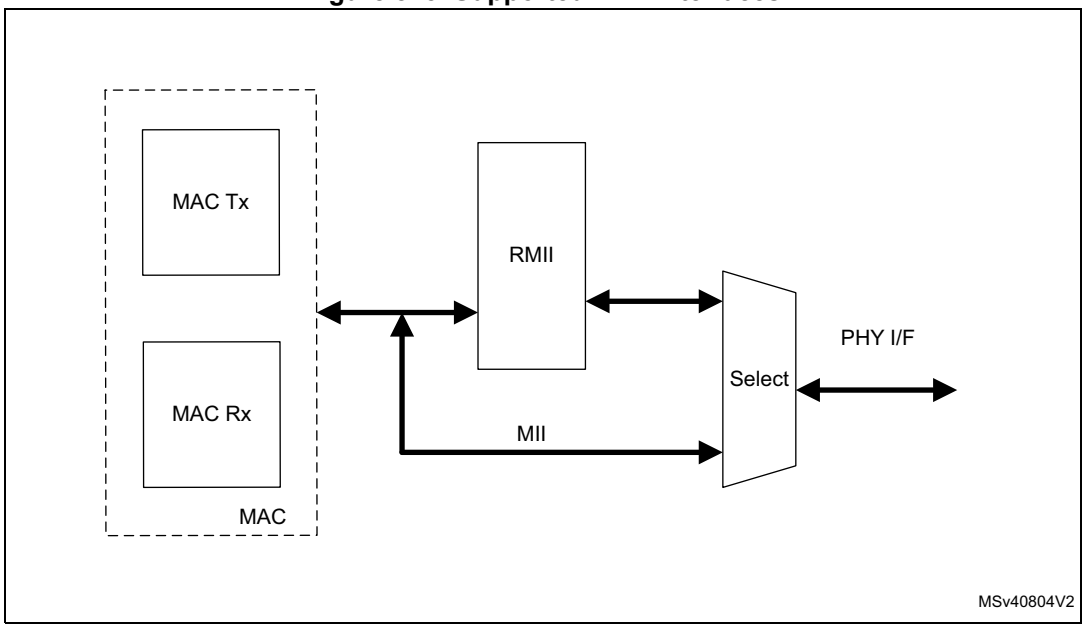
57.5.12 MAC and MMC register descriptions

Refer to *Section 57.11.4: Ethernet MAC and MMC registers*.

57.6 Ethernet functional description: PHY interfaces

The Ethernet peripheral support several PHY interfaces. The root interface is the MII one. All other interfaces are derived from it as shown in *Figure 815*.

Figure 815. Supported PHY interfaces



This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

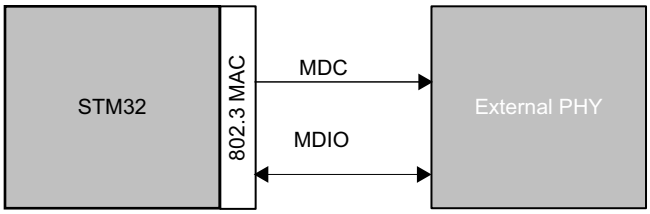
- *Station management agent (SMA)*
- *Media independent interface (MII)*
- *Reduced media independent interface (RMII)*

57.6.1 Station management agent (SMA)

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in *Figure 816*.

Figure 816. SMA Interface block



SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock (eth_hclk). The maximum operating frequency of the ETH_MDC pin is 2.5 MHz, as specified in IEEE 802.3 specifications. However, a different divider can be selected if the system supports higher clock frequencies. The division factor depends on the clock range setting through CR[3:0] in the *MDIO address register (ETH_MACMDIOAR)* register. The MDC clock is selected as follows:

Table 666. MCD clock selection

Selection	MDC clock
0000	eth_hclk/42
0001	eth_hclk/62
0010	eth_hclk/16
0011	eth_hclk/26
0100	eth_hclk/102
0101	eth_hclk/124
0110, 0111	-

The data exchange between the MAC and the PHY is performed through a three-state buffer and brought out as ETH_MDIO line connected to the PHY.

Figure 817 shows the structure of a Clause 45 MDIO packet, while *Table 667* provides a detailed description of the packet fields.

Figure 817. MDIO packet structure (Clause 45)

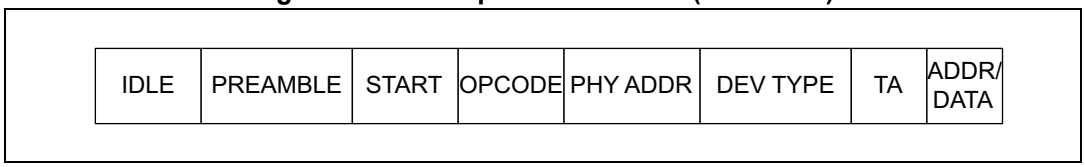
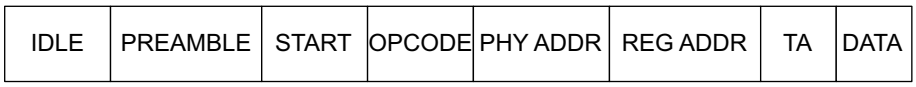


Table 667. MDIO Clause 45 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b00
OPCODE	– 0b00: Address – 0b01: Write – 0b10: Read+ Address – 0b11: Read
PHY ADDR	5-bit address select for one of 32 PHYs
DEV TYPE	5-bit device type
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	16-bit value: for an address cycle (OPCODE = 0b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data to be written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. – In address and data write cycles, the peripheral drives the ETH_MDIO line during the transfer of these 16 bits. – In read and post-read increment address cycles, the PHY drives the ETH_MDIO line during the transfer of these 16 bits.

The frame structure for Clause 22 frames is also supported. The C45E bit in *MDIO address register (ETH_MACMDIOAR)* can be programmed to enable Clause 22 or Clause 45 mode of operation. *Figure 818* shows the structure of a Clause 22 MDIO packet, while *Table 668* provides a detailed description of the packet fields.

Figure 818. MDIO packet structure (Clause 22)



MSv40806V1

Table 668. MDIO Clause 22 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b01
OPCODE	0b10 for Read and 0b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	Any 16-bit value. In a Write operation, the MAC drives ETH_MDIO. In a Read operation, the PHY drives it.

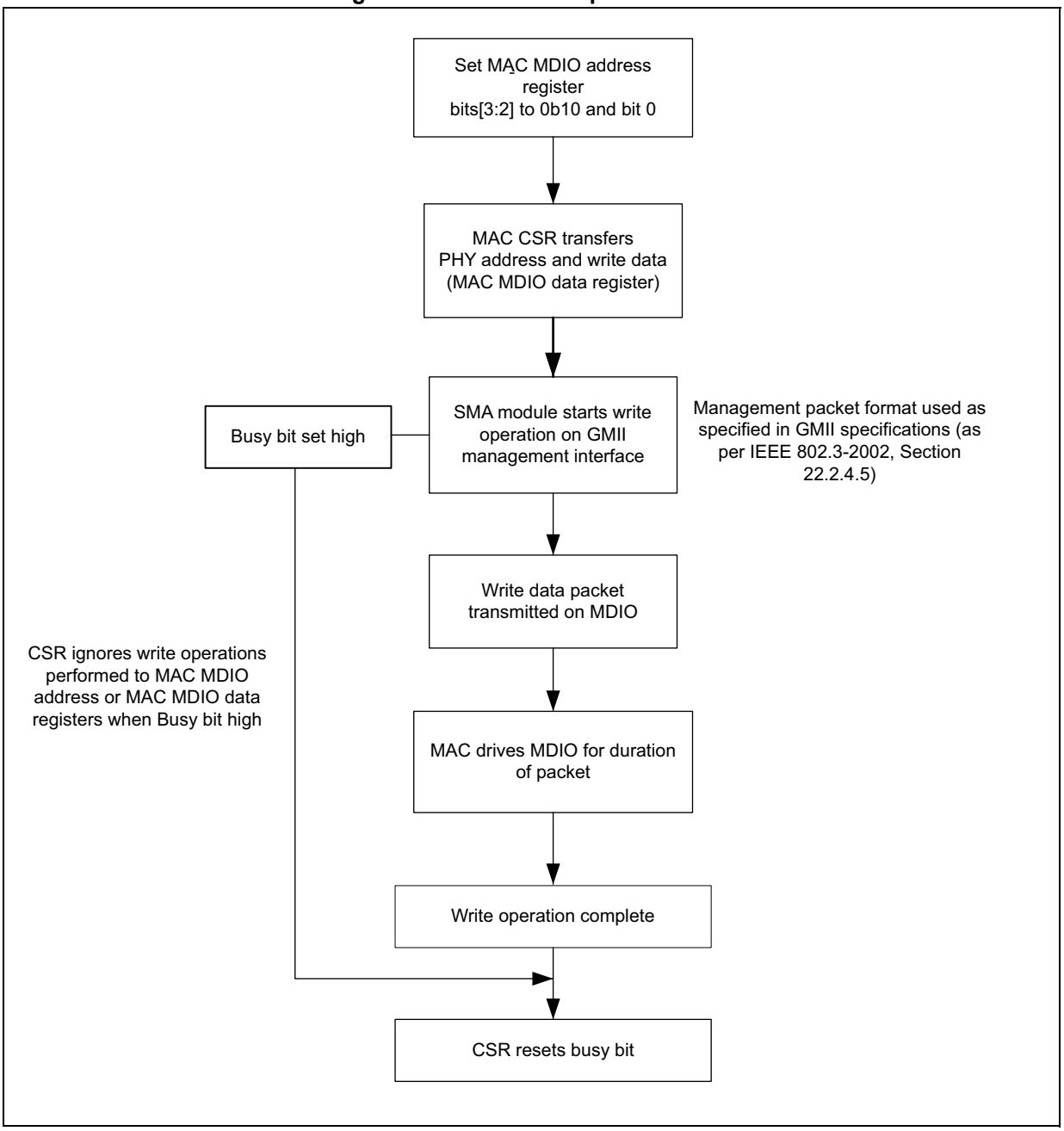
In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

MII management write operations

After the station management agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a write operation to the PHY registers.

Figure 819 illustrates the flow for a write operation from the SMA module to the PHY registers.

Figure 819. SMA write operation flow



When bits[3:2] are set to 01 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (*MDIO data register (ETH_MACMDIODR)*) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the MII management Interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations performed to the *MDIO address register (ETH_MACMDIOAR)* or the *MDIO data register*

(ETH_MACMDIODR) during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

Figure 820. Write data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

MSv40807V1

MII management read operation

When bits[3:2] are set to 11 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the MII management interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the *MDIO address register (ETH_MACMDIOAR)* or *MDIO data register (ETH_MACMDIODR)* register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the *MDIO data register (ETH_MACMDIODR)* with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

Figure 821. Read data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	10	AAAAA	RRRRR	Z0	DDD...DDD	Z

MSv40808V1

Preamble suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The peripheral provides controls to support preamble suppression. It transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the PSE bit in *MDIO address register (ETH_MACMDIOAR)*.

Trailing clocks and back-to-back transactions

The peripheral drives the ETH_MDC clock for the duration of the MDIO frame. There is no clock driven during the idle period. The trailing clock feature can be used if the PHY needs the ETH_MDC clock to be active for some cycles after the MDIO frame. The NTC[2:0] bitfield in *MDIO address register (ETH_MACMDIOAR)* allows the programming of trailing clocks from 0 to 7.

The peripheral supports back-to-back transactions which allow starting the next MDIO frame even before the trailing clocks are complete for previous MDIO frame. This feature can be enabled by setting BTB bit in *MDIO address register (ETH_MACMDIOAR)* when the trailing clock feature is also enabled. When back-to-back transactions are enabled, the GMII busy bit (GB) is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by the peripheral while trailing clocks are still on for the previous MDIO frame. When (GB) transactions are not enabled, the GMII busy bit is cleared after the trailing clocks are complete for the MDIO frame.

Interrupt for MDIO transaction completion

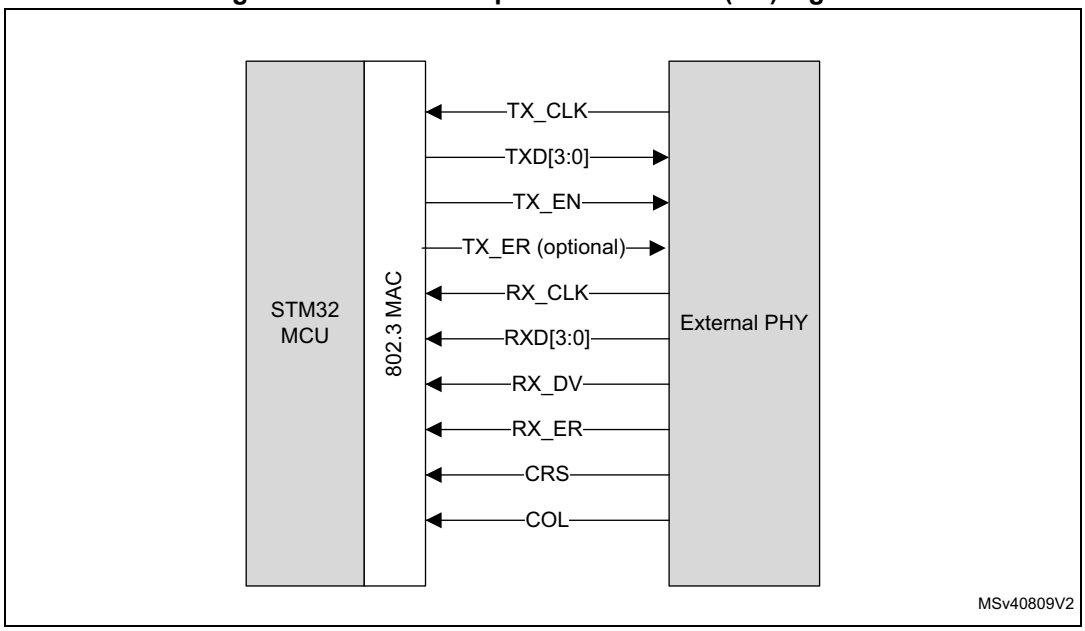
The peripheral can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy bit of the *MDIO address register (ETH_MACMDIOAR)* to know the completion of MDIO commands.

57.6.2 Media independent interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 822: Media independent interface (MII) signals*.

Figure 822. Media independent interface (MII) signals



- TX_CLK: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- TXD[3:0]: transmit data.

  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TX_ER (optional): required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- RXD[3:0]: receive data

  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is

the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.

- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.

- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.

- CRS: carrier sense.

  This signal is asserted by the PHY when either the transmit or receive medium is non idle. It is deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don't care for the MAC sublayer.

- COL: collision detection signal

  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode, the state of this signal is don't care for the MAC sublayer.

57.6.3 Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU. According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

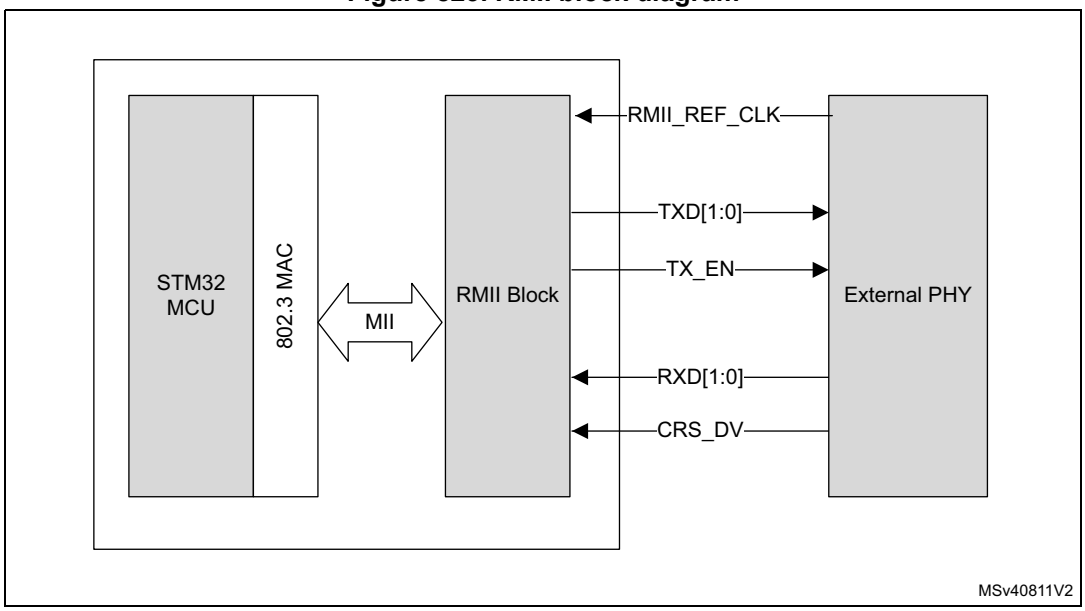
Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps operation.
- Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock references externally.

RMII block diagram

Figure 823: RMII block diagram shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

Figure 823. RMII block diagram

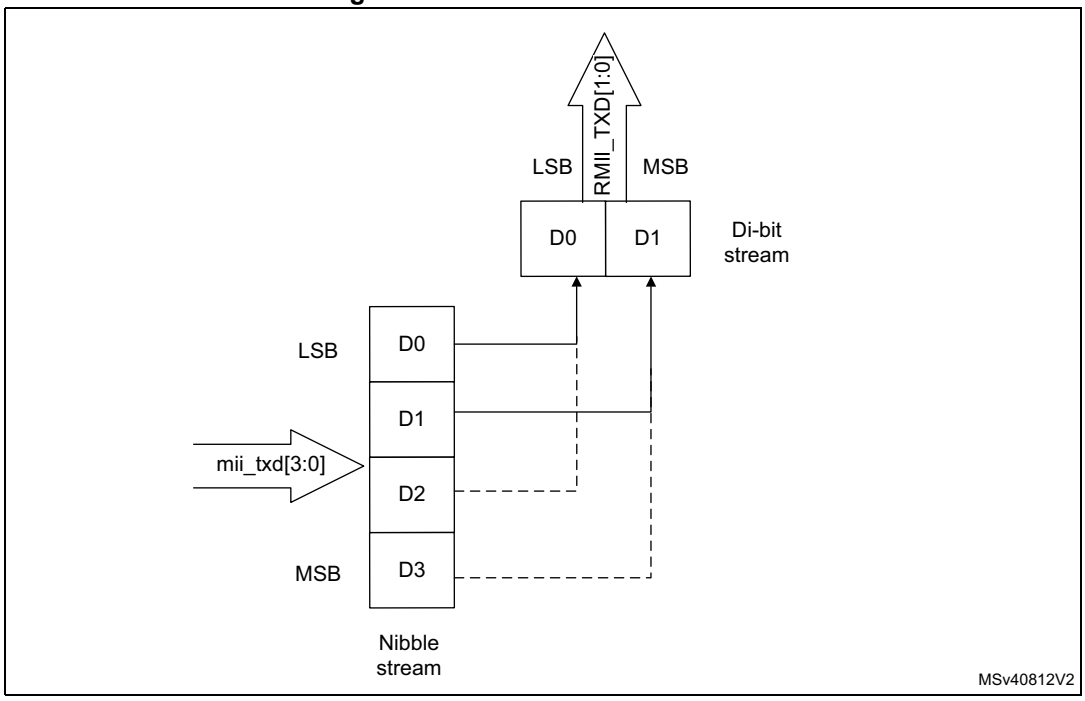


- RMII_REF_CLK: continuous 50 MHz reference clock input
- TXD[1:0]: transmit data
- TX_EN: transmit data enable.
  When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- RXD[1:0]: receive data
- CRS_DV: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.

Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in *Figure 824: Transmission bit order*. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

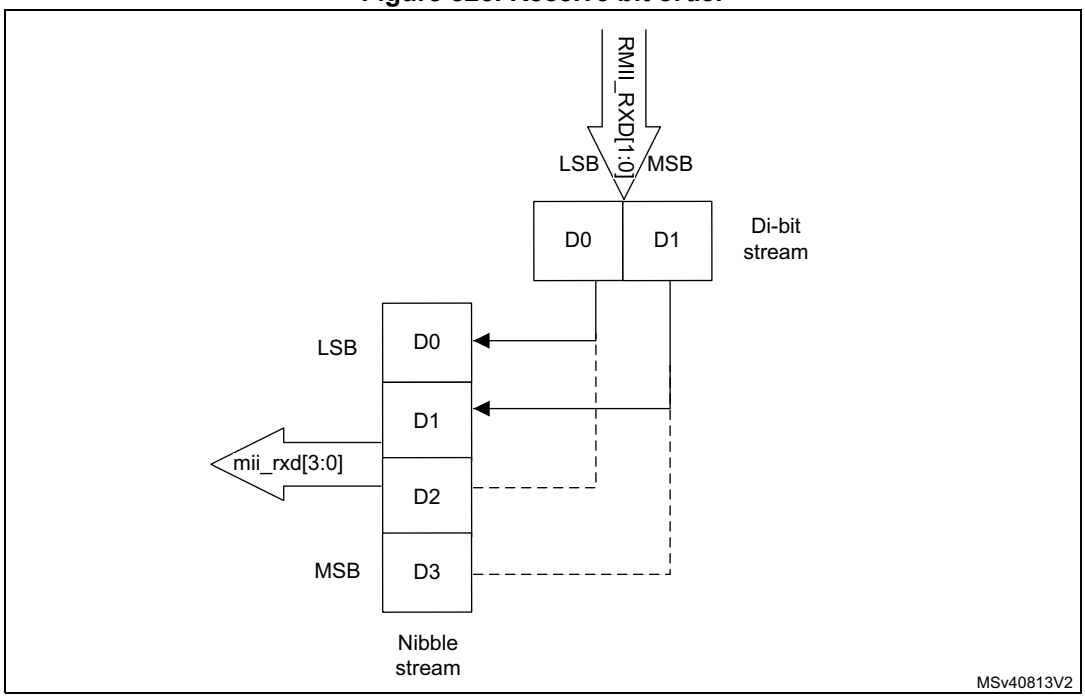
Figure 824. Transmission bit order



Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in *Figure 825: Receive bit order*. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 825. Receive bit order



57.7 Ethernet low-power modes

57.7.1 Low-power management

The Ethernet peripheral supports the following techniques to save power:

- Magic packet
- Remote wake-up

The magic packet and remove wake-up techniques are used to save power in the host system when it is in low-power mode, and has to be woken up only at the reception of specific packets from the Ethernet network. In low-power mode, the clocks on most of the host logic, along with the majority of the peripherals (except the MAC receiver logic), can be disabled. On receiving the specific packets from the network, the MAC generates the trigger to wake up the host system and come back to normal state. Refer to the power control (PWR) section of the reference manual for the list of the system operating modes that support Ethernet low-power modes.

The Energy Efficient Ethernet (EEE) mode is compliant with the IEEE 802.3az-2010 standard. It is primarily targeted to save power in the Ethernet port when there is no traffic on the line. In this mode, the host indicates to the far-end that it does not have any packets to transmit in the near future and puts the transmitter port (MAC controller, PCS and PHY layers) in low-power mode. Similarly, the receiver port can also be put in low-power mode

when the far-end host indicates that it does not have any traffic to transfer. This allows significant saving of power in the Ethernet port (mainly in the PHY) with intermittent and burst traffic profile. The triggering of entry and exit out of the EEE mode is controlled by the MAC and is supported within the peripheral.

Simultaneous operation of the EEE mode along with any or both the other power saving modes is also supported.

Description of magic packet mode

This section describes how to save power through magic packet detection.

Note: The magic packet feature is based on the magic packet technology white paper from Advanced Micro Device (AMD).

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the magic-packet-based power saving mode, the reception of a valid magic packet by the MAC receiver triggers an exit from low-power mode. The MAC enters power saving mode when PWRDWN bit of *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the magic-packet-based power saving mode is enabled by setting the MGKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The magic packet contains a unique pattern at any offset after the Destination address, Source address, and Length/Type fields. In addition to the unique pattern matching, the MAC receiver also checks for the following, to detect the received packet as a valid magic packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

Magic packet data format

The content of the unique pattern in magic packets is described as follows:

- Six bytes of all-ones (0xFF FF FF FF FF FF) called synchronization stream. There can be more than six bytes of 0xFF, but only the last six are considered.
- The synchronization stream is immediately followed by 16 repetitions of the Destination address field of the packet (*MAC Address 0 high register (ETH_MACA0HR)* and the *MAC Address 0 low register (ETH_MACA0LR)*) or multicast/broadcast address.
- No break or interruption between synchronization stream and first repetition of Destination address field or within its 16 repetitions.

If the MAC address of a node is 0x00 11 22 33 44 55, the MAC scans for the following data sequence:

Destination Address Source Address Length/Type…… FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
…CRC

Description of remote wake-up packet mode

This section describes the power saving mode based on remote wake-up packet.

Note: The remote wake-up packet feature implementation is based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the remote-wake-up-magic-packet-based power saving mode, the reception of expected remote wake-up packet by the MAC receiver triggers the exit from low-power mode. The MAC enters power saving mode when PWRDWN bit in *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the remote-wake-up-magic-packet-based power saving mode is enabled by programming RWKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The MAC implements a filter lookup table (programmed through *Remote wake-up packet filter register (ETH_MACRWKPFR)* in which CRC, offset, and byte mask of the pattern embedded in remote wake-up packet and the filter operation commands are programmed.

The pattern embedded in the remote wake-up packet is located at any offset after the Destination address and Source address fields. In addition to the CRC match for the pattern, the MAC receiver also checks the following, to detect the received packet as a valid remote wake-up packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

When a valid remote wake-up packet is received, the MAC receiver sets the RWKPRCVD bit in *PMT control status register (ETH_MACPCSR)* and triggers the interrupt on pmt_intr_o output port. The PMTIS bit in *Interrupt status register (ETH_MACISR)* is set when power-gating is not enabled in low-power mode. An interrupt is triggered to the application on the sbd_intr_o output port when interrupt is enabled (PMTIE bit in *Interrupt enable register (ETH_MACIER)* is set) and CSR clock is not gated off in low-power mode.

Remote wake-up packet filters

When the remote-wake-up-based power saving mode is enabled, four remote wake-up filters can be selected. The structure of the remote wake-up filters is shown in *Table 669: Remote wake-up packet filter register*.

Table 669. Remote wake-up packet filter register

ETH_MACRWKPFR value	Field							
0	Filter 0 byte mask							
1	Filter 1 byte mask							
2	Filter 2 byte mask							
3	Filter 3 byte mask							
4	Reserved	Filter 3 command	Reserved	Filter 2 command	Reserved	Filter 1 command	Reserved	Filter 0 command
5	Filter 3 offset		Filter 2 offset		Filter 1 offset		Filter 0 offset	
6	Filter 1 CRC - 16				Filter 0 CRC - 16			
7	Filter 3 CRC - 16				Filter 2 CRC - 16			

The remote wake-up filter fields are described in *Table 670: Description of the remote wake-up filter fields*.

Table 670. Description of the remote wake-up filter fields

Register	Description
Filter *i* Byte mask	The filter *i* byte mask register defines the bytes of the packet that are examined by filter *i* (0, 1, 2 or 3) to determine whether or not a packet is a wake-up packet. – The MSB (31st bit) must be zero. – Bit j[30:0] is the byte mask. – If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter *i* Offset + j of the incoming packet; otherwise Filter *i* Offset + j is ignored.
Filter *i* Command	The 4-bit filter *i* command controls the filter *i* operation. – Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. – Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value. Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2". – Bit 1 (And_Previous) implements the Boolean logic(1). When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. – Bit 0 is the enable for filter *i*. If Bit 0 is not set, filter *i* is disabled.

Table 670. Description of the remote wake-up filter fields (continued)

Register	Description
Filter *i* Offset	This filter *i* offset register defines the offset (within the packet) from which the filter *i* examines the packets. – This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. – The minimum allowed offset is 12, which refers to the 13th byte of the packet. – The offset value 0 refers to the first byte of the packet.
Filter *i* CRC-16	This filter *i* CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block. – The 16-bit CRC calculation uses the following polynomial: $G(x) = x^{16} + x^{15} + x^2 + 1$ Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following: – 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation. – 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

1. The And_Previous bit setting is applicable within a set of four filters.
Setting And_Previous bit of a filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of four filters has no effect. For example, setting And_Previous bit of Filter 0 has no effect.
If And_Previous bit is set for a given filter to form an AND chained filter, the AND chain breaks when it finds a disabled filter. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting Filter 2 And_Previous bit has no effect, only Filter 1 result ORed with Filter 3 result is considered.
If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, then a remote wake-up frame does not pass the AND chained filter as a remote wake-up frame cannot be of both unicast and multicast address types.

The remote wake-up filter registers are implemented as eight indirect access registers (wkuppktfilter_reg#i) for four remote wake-up filters, and accessed by the application through *Remote wake-up packet filter register (ETH_MACRWKPFR)*. The entire set of wkuppktfilter_reg registers must be written to program the remote wake-up filters. The wkuppktfilter_reg register is programmed by sequentially writing the eight register values in *Remote wake-up packet filter register (ETH_MACRWKPFR)* for wkuppktfilter_reg0 to wkuppktfilter_reg3, respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in RWKPTR field of *PMT control status register (ETH_MACPCSR)*.

Note: If the Remote wake-up packet filter register (ETH_MACRWKPFR) is accessed in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when the CPU accesses Lane 3.

When Remote wake-up packet filter register (ETH_MACRWKPFR) is written, the content is transferred from CSR clock domain to PHY receive clock domain after the write operation. There should not be any further write to the Remote wake-up packet filter register (ETH_MACRWKPFR) until the first write is updated in PHY receive clock domain. Otherwise, the second write operation does not get updated to the PHY receive clock domain. Therefore, the delay between two write operations to the Remote wake-up packet filter register (ETH_MACRWKPFR) should be at least 4 cycles of the PHY receive clock.

PMT interrupt

The PMT interrupt signal is asserted when a valid remote wake-up packet is received.

Table 671 lists the remote wake-up scenarios in which PMT interrupt is generated.

Table 671. Remote wake-up packet and PMT interrupt generation(1)

Filter i Command			Frame Type and CRC Status		Interrupt Generation
CAST	INV	EN	Received Frame Cast Type	CRC Status	RWK INTR
0	0	1	Unicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
0	1	1	Unicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	0	1	Multicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	1	1	Multicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated

1. In all other combinations, the Remote Wake-up packet is not detected and PMT interrupt is not generated.

In addition to sbd_intr_o signal, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the power-down mode. It is ORed with lpi_intr_o signal (see *Section : LPI interrupt*) and tied to the EXTI peripheral.

As the pmt_intr_o signal is generated in the PHY Rx clock domain, it is not cleared immediately when the *PMT control status register (ETH_MACPCSR)* is read. This is because the resultant clear signal has to cross to the PHY Rx clock domain, and then clear the interrupt source. This delay is at least 4 clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode. When the application clears the PWRDWN bit in *Remote wake-up packet filter register (ETH_MACRWKPFR)*, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

Power-down sequence

The software must perform the following tasks to initiate the power-down sequence:

- Disable the Transmit DMA (if applicable) by clearing the ST bit of the *Channel transmit control register (ETH_DMACTXCR)*.
- Wait for any previous frame transmissions to complete. You can check this by reading TFCSTS[1:0] and TPESTS bits in *Debug register (ETH_MACDR)* and TXQSTS bit in *Tx queue debug register (ETH_MTLTXQDR)* of all MTL Tx Queues.
- Disable the MAC transmitter and MAC receiver by clearing TE and RE bits in *Operating mode configuration register (ETH_MACCR)*.
- Wait till the Receive DMA empties all frames from the Rx FIFO. You can check this by reading PRXQ[13:0] in *Rx queue debug register (ETH_MTLRXQDR)* of all Rx Queues. If these bits are zero, it indicates that the Rx FIFO is empty.
- Configure the magic packet (MGKPKTEN) and/or remote wake-up (RWKPKTEN) detection in the *PMT control status register (ETH_MACPCSR)*.
- Set bit 31 (ARPEN) in the *Operating mode configuration register (ETH_MACCR)*.
- Enable the MAC receiver by setting RE bit and then set PWRDWN bit in the *PMT control status register (ETH_MACPCSR)* to initiate the power-down sequence in MAC.

Note: If the feature is enabled and the MAC transmitter is in the LPI mode when it is put into the power-down mode, then the MII interface gets clamped to assert the LPI pattern. If the MAC transmitter is not in the LPI mode when it is put into the power-down mode, the GMII or MII interface gets clamped to all-zero.

Power-up sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The power-up sequence is as follows:

- The MAC asserts pmt_intr_o. When only clock-gating is employed in low-power mode, the pmt_intr_o signal can be used to start the clocks that were gated-off after entering low-power mode.
- The software performs the following tasks:
  - De-assert the pmt_intr_o by reading the *PMT control status register (ETH_MACPCSR)*.
  - Perform a write operation (with reset values) to the *PMT control status register (ETH_MACPCSR)* and the *Remote wake-up packet filter register (ETH_MACRWKPFR)* so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
  - Perform write operations to the *Operating mode configuration register (ETH_MACCR)*, *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)* to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver is on even though the Receive Enable bit is set to 0.

After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

57.7.2 Energy Efficient Ethernet (EEE)

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The

EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps. The peripheral supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows saving power by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit path functions

The transmit path functions include tasks that the MAC must perform to make the PHY enter the LPI state.

In the transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER LST[9:0] bitfield of *LPI timers control register (ETH_MACLTCR)*. The PHY Link Status PLS bit of the *LPI control and status register (ETH_MACLCSR)* indicates the link status of the PHY.

Note: The EEE feature is not supported when the MAC is configured to use the RMII.

According to the standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbps)
4. Updates the status (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

To bring the PHY out of the LPI state, that is when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:

   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the *LPI timers control register (ETH_MACLTCR)*.
3. Updates the LPI exit status (TLPIEX bit of the *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Figure 826 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Note: The MAC does not stop the TX_CLK clock. The application can stop this clock (as shown in Figure 826) if the PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o

signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay, and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal depends on the LPITCSE bit of the LPI control and status register (ETH_MACLCSR) and can be done automatically as shown on Figure 827.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.

If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.

If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

Figure 826. LPI transitions (Transmit, 100 Mbds)

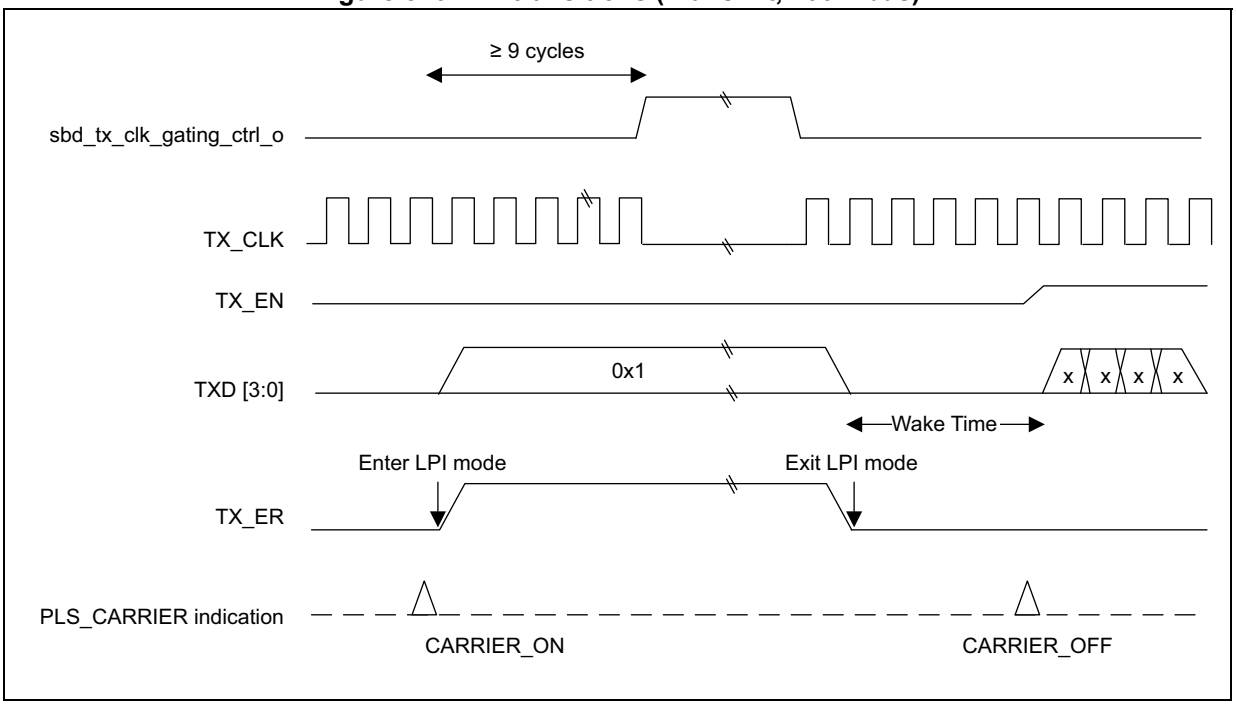
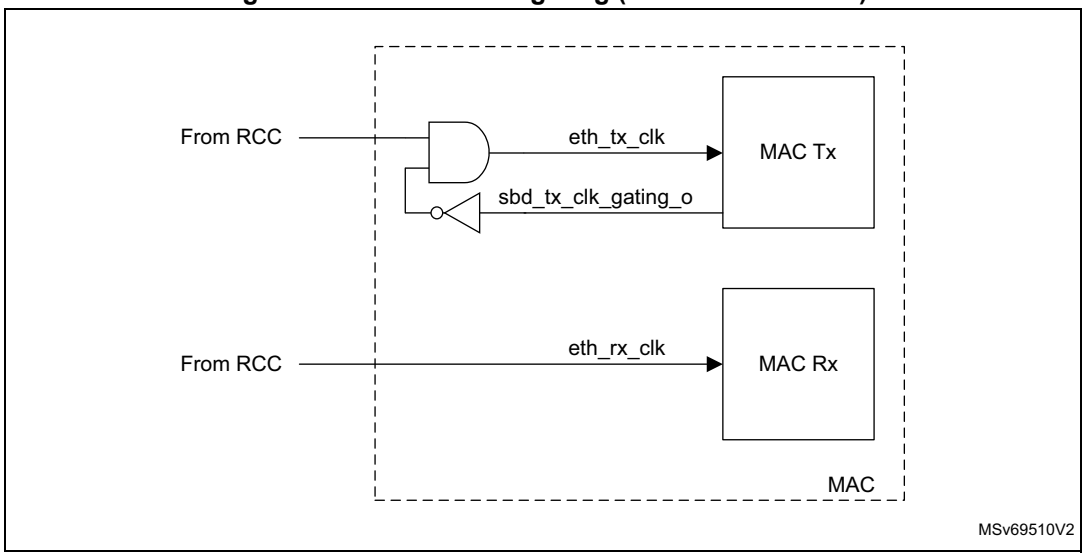


Figure 827. LPI Tx clock gating (when LPITCSE = 1)



Automated entry/exit from LPI mode in transmit path

The MAC transmitter can be programmed to enter and exit LPI Idle mode automatically based on whether it is Idle for a specific period of time or has a packet to transfer. These modes are enabled and controlled by the *LPI control and status register (ETH_MACLCSR)*.

When LPITXA and LPIEN of *LPI control and status register (ETH_MACLCSR)* are set, the MAC transmitter enters LPI Idle state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI Idle state and clears the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when LPITE is also set, the MAC transmitter enters LPI Idle state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in *LPI entry timer register (ETH_MACLETR)*. In this mode also, the MAC transmitter exits the LPI Idle state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI Idle state is possible without any software intervention when the MAC becomes idle again.

When both LPITE and LPITXA bits are cleared, the application can directly control the entry and exit of LPI Idle state by programming the LPIEN bit.

Receive path functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[3:0] to 0x1 (for 100 Mbps).
3. The PHY de-asserts RX_DV.
4. The MAC updates the RLPIEN bit of the *LPI control and status register (ETH_MACLCSR)* and immediately generates an interrupt.

Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.

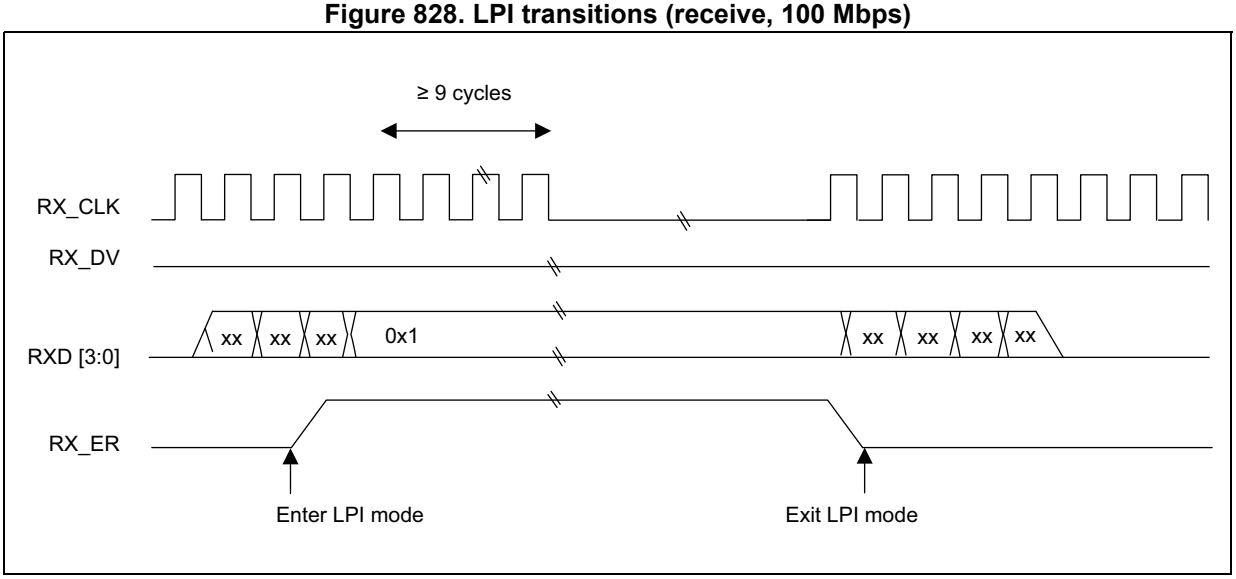
If the duration between the end of the current Rx LPI pattern and the start of the next Rx LPI pattern is very short (that is, less than two Rx clock cycles), then the MAC exits and enters again the Rx LPI mode. The MAC does not trigger the Rx LPI Exit and Entry interrupts.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the *LPI control and status register (ETH_MACLCSR)* and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 828 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 828. LPI transitions (receive, 100 Mbps)



Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in Figure 828.

If the MAC is in LPI mode and the application issues a soft reset or hard reset, the MAC receiver exits from LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver enters again the LPI state.

If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.

When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.

LPI timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

The following LPI timers are loaded with the respective values from the *LPI timers control register (ETH_MACLTCR)* and *LPI entry timer register (ETH_MACLETR)*:

- LPI LS TIMER

  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up.

  This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the LST[9:0] bitfield in the *LPI timers control register (ETH_MACLTCR)*. The LPI LS TIMER is 10-bit wide. The software can program up to 1023 milliseconds.
- LPI TW TIMER

  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of *LPI timers control register (ETH_MACLTCR)*. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.

  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 micro seconds.
- LPI AUTO ENTRY TIMER

  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in *LPI control and status register (ETH_MACLCSR)* is set.

LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt sbd_intr_o is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the *LPI control and status register (ETH_MACLCSR)*.

When the MAC exits the Rx LPI state, then in addition to the sbd_intr_o, the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. The lpi_intr_o signal can be used to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the LPI state. It is ORed with pmt_intr_o signal (see *Section : PMT interrupt*) and tied to the EXTI peripheral.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the *LPI control and status register (ETH_MACLCSR)* is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode.

Programming guidelines for Energy Efficient Ethernet

For detailed guidelines on the programming guidelines, see *Section 57.9.11: Programming guidelines for Energy Efficient Ethernet (EEE) on page 2797*.

57.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the *Interrupt status register (ETH_DMAISR)* register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMACSR: Channel Status (see *Channel status register (ETH_DMACSR)*))
- ETH_MTLISR: Interrupt Status (see *Interrupt status register (ETH_MTLISR)*)
- ETH_MACISR: Interrupt Status (see *Interrupt status register (ETH_MACISR)*)

57.8.1 DMA interrupts

Interrupt registers description

The ETH_DMACSR: Channel Status register (see *Channel status register (ETH_DMACSR)*) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMACIER: Channel Interrupt Enable register (see *Channel interrupt enable register (ETH_DMACIER)*) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts.They are indicated by Bits[15:14] of ETH_DMACSR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the *Interrupt status register (ETH_DMAISR)* for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of *Interrupt status register (ETH_DMAISR)* are cleared.

Periodic scheduling of Transmit and Receive interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in *Table 675: TDES2 normal descriptor (read format)*) once for every “required” number of packets to be transmitted.
2. Similarly, set the IOC (RDES3[30] in*Table 688: RDES3 normal descriptor (read format)*) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.

In addition to above, an interrupt timer (ETH_DMACRXIWTR: Channel Rx Interrupt Watchdog Timer) is given for flexible control and periodic scheduling of Receive interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in *Table 688: RDES3 normal descriptor (read format)*) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMACIER register (see *Channel interrupt enable register (ETH_DMACIER)*). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Channel transfer complete interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in the Channel Status register (*Channel status register (ETH_DMACSR)*). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30])).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (*Channel interrupt enable register (ETH_DMACIER)*).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bits[17:16]) in the ETH_DMAMR register (*DMA mode register (ETH_DMAMR)*). *Table 672* explains the behavior of the Transfer Complete interrupt.

Table 672. Transfer complete interrupt behavior

Interrupt Mode	Behavior of TI/RI and interrupt signal
INTM=0	The TI/RI status signals are set whenever the Transfer complete event is detected. These bits are cleared whenever the software driver writes 1 to these bits. The interrupt signal is asserted whenever the corresponding interrupts are also enabled in ETH_DMACIER register.
INTM=1	The TI/RI is set as explained above. However, the interrupt is not asserted for any RI/TI event.
INTM=2	The RI/TI status bits are set whenever the Transfer Complete event is detected and are reset whenever software driver clears them by writing 1. However, if another Transfer complete event is detected before it is cleared (serviced) by the software, then these status bits are automatically set again. However, the interrupt is not generated based on TI/RI.

57.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

The register *Interrupt status register (ETH_MTLISR)* report the queue number responsible for the event. ETH_MTLQICSR: Queue Interrupt Control Status must be read for event description.

The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MTLISR)* register.

MTL interrupt signal is driven by one of these events:

- Receive Queue Overflow Interrupt
- Transmit Queue Underflow

57.8.3 MAC Interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MACISR)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:

- Receive Status Interrupt
- Transmit Status Interrupt
- Timestamp Interrupt Status
- MMC Interrupt Status
  - MMC Receive Checksum Offload Interrupt Status
  - MMC transmit interrupt status
  - MMC Receive interrupt status
- LPI Interrupt Status
- PMT Interrupt Status
- PHY Interrupt

Note: Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wake-up event detection at EXTI level.

57.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see *Section 57.9.1*)
- MTL initialization (see *Section 57.9.2*)
- MAC initialization (see *Section 57.9.3*)
- Performing normal receive and transmit operation (see *Section 57.9.4*)
- Stopping and starting transmission (see *Section 57.9.5*)
- Programming guidelines for MII link state transitions (see *Section 57.9.8*)
- Programming guidelines for IEEE 1588 timestamping (see *Section 57.9.9*)
- Programming guidelines for Energy Efficient Ethernet (see *Section 57.9.11*)
- Programming guidelines for flexible pulse-per-second (PPS) output (see *Section 57.9.12*)
- Programming guidelines for IEEE 1588 auxiliary snapshot (see *Section 57.9.13*)
- Programming guidelines for TSO (see *Section 57.9.14*)
- Programming guidelines for VLAN filtering on the receiver (see *Section 57.9.15*)

57.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of *DMA mode register (ETH_DMAMR)*).
2. Wait for the completion of the reset process (poll bit 0 of the *DMA mode register (ETH_DMAMR)*, which is cleared when the reset operation is completed).
3. Program the following fields to initialize the *System bus mode register (ETH_DMASBMR)*:
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values in case of AHB bus interface.
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to *Section 57.10: Descriptors*.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMACTXRLR and ETH_DMACRXRLR registers (see *Channel Tx descriptor ring length register (ETH_DMACTXRLR)* and *Channel Rx descriptor ring length register (ETH_DMACRXRLR)*). The programmed ring length must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (*Channel Tx descriptor list address register (ETH_DMACTXDLAR)*, *Channel Rx descriptor list address register (ETH_DMACRXDLAR)*). In addition, program the transmit and receive tail pointer registers that inform the DMA about the available descriptors (see *Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)* and *Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)*).
7. Program ETH_DMACCR, ETH_DMACTXCR and ETH_DMACRXCR registers (see *Channel control register (ETH_DMACCR)*, *Channel transmit control register*

(ETH_DMACTXCR) and *Channel receive control register (ETH_DMACRXCR)*) to configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ[13:0] for RxDMA, and so on.

8. Enable the interrupts by programming the ETH_DMACIER register (see *Channel interrupt enable register (ETH_DMACIER)*).
9. Start the Receive and Transmit DMAs by setting SR (bit 0) of *Channel receive control register (ETH_DMACRXCR)* and ST (bit 0) of the ETH_DMACTXCR (see *Channel transmit control register (ETH_DMACTXCR)*).

57.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the following fields to initialize the operating mode in *Tx queue operating mode register (ETH_MTLTXQOMR)*.
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) if the Threshold mode is used.
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue 0.
   c) Transmit Queue Size (TQS).
2. Program the following fields to initialize the operating mode in the ETH_MTLRXQOMR register (see *Rx queue operating mode register (ETH_MTLRXQOMR)*):
   a) Receive Store and Forward (RSF) or RTC if Threshold mode is used.
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP).
   d) Receive Queue Size (RQS).

57.9.3 MAC initialization

The MAC configuration registers establish the operating mode of the MAC. If possible, these registers must be initialized before initializing the DMA. The following MAC Initialization operations can also be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: *MAC Address x low register (ETH_MACAxLR)*, *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x high register (ETH_MACAxHR)*.
2. Program the following fields to set the appropriate filters for the incoming frames in the *Packet filtering control register (ETH_MACPFR)*:
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the *Tx Queue flow control register (ETH_MACQTXFCR)*:
   a) Pause time and other Pause frame control bits.
   b) Transmit Flow control bits.

c) Flow Control Busy.

4. Program the *Interrupt enable register (ETH_MACIER)* as required, if it is applicable for your configuration.
5. Program the appropriate fields in the *Operating mode configuration register (ETH_MACCR)* register. For example, Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in *Operating mode configuration register (ETH_MACCR)* register to start the MAC transmitter and receiver.

To support Jumbo Transmit/Receive packets, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JE bit to 1.
  b) Set JD and WD bits to 0 to avoid giant packet error reporting.
  c) Set GPSLCE bit to 1
  d) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to a value > 9026

To support Transmit/Receive packets, up to 16K, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JD and WD bits to 1 to avoid giant packet error reporting.
  b) Set GPSLCE bit to 1.
  c) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to 16383.

57.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the descriptors and writing the ETH_DMACTXDTPR (see *Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)* and ETH_DMACRXDTPR (see *Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)*).
4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see *Channel current application transmit descriptor register (ETH_DMACCATXDR)* and *Channel current application receive descriptor register (ETH_DMACCARXDR)*).
5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see *Channel current application transmit descriptor register (ETH_DMACCATXDR)* and *Channel current application receive descriptor register (ETH_DMACCARXDR)*).

57.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMACTXCR register (see *Channel transmit control register (ETH_DMACTXCR)*).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue debug register (ETH_MTLTXQDR)* (TRCSTS[1:0] is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing RE and TE bits of the *Operating mode configuration register (ETH_MACCR)* Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of *Tx queue debug register (ETH_MTLTXQDR)*, PRXQ=0 and RXQSTS[1:0] = 00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in *Tx queue debug register (ETH_MTLTXQDR)* and RXQSTS[1:0] is set to 00.
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

Note: Do not change the configuration (such as duplex mode, speed, port, or loopback) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active.

Similarly, do not change the DMA-related configuration when Transmit and Receive DMA are active.

57.9.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in Suspend state, as explained below:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to Suspend state due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of an RxFIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into Stop state.

57.9.7 Programming guidelines for switching the AHB clock frequency

To dynamically change the AHB clock frequency (without applying soft reset or hard reset), follow these steps:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. When the frame transmissions is complete, the Tx FIFO becomes empty and the Tx DMA enters Stop state. The Tx FIFO status is given in the *Tx queue debug*

register (ETH_MTLTXQDR) and the status of DMA is given in *Debug status register (ETH_DMADSR)*.

2. Disable the MAC transmitter and the MAC receiver by clearing the appropriate bits in *Operating mode configuration register (ETH_MACCR)*.
3. Disable the Receive DMA (if applicable) after making sure that the data in the Rx FIFO is transferred to the system memory. The Rx FIFO empty status is given in *Rx queue debug register (ETH_MTLRXQDR)*.
4. Ensure that the application does not perform any register read or write operation.
5. Change the frequency of the AHB clock.
6. Enable the MAC transmitter or the MAC receiver and the Transmit or Receive DMA. These steps ensure that no valid data is present in the Tx FIFO or Rx FIFO at the time of clock frequency switching and prevent any data corruption.

57.9.8 Programming guidelines for MII link state transitions

Transmit and Receive clocks are running when the link is down

Complete the following steps when the link is down while the Transmit and Receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of *Channel control register (ETH_DMACCR)*.
2. Disable the MAC receiver by clearing RE bit of *Operating mode configuration register (ETH_MACCR)*.
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue debug register (ETH_MTLTXQDR)* (TRCSTS[1:0] is not 01).

   or

   Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing TE bit of the *Operating mode configuration register (ETH_MACCR)* Register.
5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in *Tx queue debug register (ETH_MTLTXQDR)* and RXQSTS[1:0] to 00 in *Rx queue debug register (ETH_MTLRXQDR)*).
6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.
7. Restart the operation by starting the Tx DMA. Then enable the MAC transmitter and receiver.

   The Rx DMA does not need to be enabled: since the receiver is disabled, there are no data in the Rx FIFO.

Transmit and Receive clocks are stopped when the link is down

Complete the following steps when the link is down and the Transmit and Receive clocks are stopped:

1. Disable the MAC transmitter and receiver by clearing RE and TE bits in the *Operating mode configuration register (ETH_MACCR)*. This does not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the *Debug register (ETH_MACDR)* (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC transmitter is stopped.
4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC transmitter and receiver by setting RE and TE bits.

57.9.9 Programming guidelines for IEEE 1588 timestamping

Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the *Timestamp control Register (ETH_MACTSCR)*. However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 12 of *Interrupt enable register (ETH_MACIER)*.
2. Set bit 0 of *Timestamp control Register (ETH_MACTSCR)* to enable timestamping.
3. Program *Subsecond increment register (ETH_MACSSIR)* based on the PTP clock frequency.
4. If you use the Fine Correction method, program *Timestamp addend register (ETH_MACTSAR)* and set bit 5 of *Timestamp control Register (ETH_MACTSCR)*.
5. Poll the *Timestamp control Register (ETH_MACTSCR)* until bit 5 is cleared.
6. Program bit 1 of *Timestamp control Register (ETH_MACTSCR)* to select the Fine Update method (if required).
7. Program *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)* with the appropriate time value.
8. Set bit 2 in *Timestamp control Register (ETH_MACTSCR)*.

   The timestamp counter starts as soon as it is initialized with the value written in the timestamp update registers. If one-step timestamping is required:

   a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.

   b) Program *Timestamp ingress asymmetric correction register (ETH_MACTSIACR)* to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Note: If timestamp operation is disabled by clearing bit 0 of Timestamp control Register (ETH_MACTSCR), repeat all these steps to restart the timestamp operation.

System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (*System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*).
2. Set bit 3 (TSUPDT) of the *Timestamp control Register (ETH_MACTSCR)*.

   The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in *Section : System time register module*, calculate at which rate you intend to increment or decrement the system time.
2. Update the *Timestamp addend register (ETH_MACTSAR)* with the new value and set bit 5 of the *Timestamp control Register (ETH_MACTSCR)* Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*.
5. Enable the Timestamp interrupt in bit 12 of *Interrupt enable register (ETH_MACIER)*.
6. Set bit 4 in Register *Timestamp control Register (ETH_MACTSCR)*.
7. When this trigger generates an interrupt, read *Interrupt status register (ETH_MACISR)*.
8. Reprogram *Timestamp addend register (ETH_MACTSAR)* with the old value and set bit 5 again.

57.9.10 Programming guidelines for PTP offload feature

Programming guidelines to enable automatic periodic generation of PTP sync messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program SNAPTYPSEL, TSMSTRENA, and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 1, and 1 respectively, to configure the node as Ordinary or Boundary Master (1, 1, and 1 for Transparent Master).
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Sync message.
3. Program the ASYNCEN bit of *PTP Offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP*

Source port identity 2 register (ETH_MACSPI2R) to send in egress PTP Sync message.

5. Program the LSI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Sync messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Sync message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Sync message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable periodic generation of PTP Pdelay_Req messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 0, and 1 respectively to configure the node as Transparent Slave (1, 1, and 1 for Transparent Master OR 3, X, and X for Peer-to-Peer Transparent).
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Pdelay_Req message.
3. Program the APDREQEN bit of *PTP Offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Pdelay_Req messages.
4. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Pdelay_Req message.
5. Program the LMPDRI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Pdelay_Req messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Pdelay_Req message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Pdelay_Req message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Master mode

Follow these steps to enable the generation of PTP response messages for Ordinary or Boundary Master mode (Periodic PTP Sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the DRSYNCR and LSI fields in *Log message interval register (ETH_MACLMIR)*. The sum of both fields is updated in logMinDelayReqInterval field of PTP Delay_Resp message.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Slave mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Slave mode (PTP Delay_Req message generated in response to PTP Sync message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 0, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the DRSYNCR field in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Slave mode

Follow these steps to enable generation of PTP response messages for Transparent Slave mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated)

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync or

Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.

4. Program the DRSYNCR and LMPDRI fields in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Master mode

Follow these steps to enable generation of PTP response messages for Transparent Master mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 1, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
4. Program the DRSYNCR, LSI and LMPDRI fields in *Log message interval register (ETH_MACLMIR)*, the sum of DRSYNCR and LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Peer-to-Peer Transparent mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):

1. Program the SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control Register (ETH_MACTSCR)* to 3, X, and X respectively.
2. Program the PTOEN bit and DN field of *PTP Offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP Source Port Identity 0 Register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
4. Program the LMPDRI field in *Log message interval register (ETH_MACLMIR)* to indicate periodicity of the PTP Pdelay_Req messages
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

57.9.11 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-power idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol.

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).
3. Program bits 25 to 16 and bits 15 to 0 in *LPI timers control register (ETH_MACLTCR)*.
4. Read the PHY link status by using the MDIO interface and update bit 17 of *LPI control and status register (ETH_MACLCSR)*.

   Update *LPI control and status register (ETH_MACLCSR)* accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program *One-microsecond-tick counter register (ETH_MAC1USTCR)* as per the frequency of the clock used for accessing the CSR slave port.
6. Program the LPIET bit in the *LPI entry timer register (ETH_MACLETR)* with the IDLE time for which the MAC should wait before entering the LPI state on its own.

7. Set LPITE and LPITXA (bits 20 to 19) of *LPI control and status register (ETH_MACLCSR)* to enable LPI auto-entry and MAC auto-exit from LPI state.
8. Set bit 16 of *LPI control and status register (ETH_MACLCSR)* to put the MAC transmitter in LPI state.

   The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.
9. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.
10. The MAC transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.
11. Reset LPIEN bit if the application needs to override the auto-entry/exit modes and directly exit the MAC transmitter from LPI state.

Note: To make sure the MAC enters the LPI state only after the transmission of all the queued frames in the Tx FIFO is complete, set LPITXA bit in LPI control and status register (ETH_MACLCSR).

To switch off the CSR clock or power to the rest of the system during the LPI state, wait for the TLPIEN interrupt of LPI control and status register (ETH_MACLCSR) to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

Gating Off the CSR Clock in the LPI mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

Gating off the CSR clock in the Rx LPI mode

The following operations are performed when the MAC receives the LPI pattern from the PHY:

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status (RLPIEN interrupt of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, the CSR clock can be gated off. If the MAC TX is not in LPI mode when the CSR clock is gated off, the events on the MAC transmitter do not get reported or updated in the CSR. To restore the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when *LPI control and status register (ETH_MACLCSR)* is read.

Gating off the CSR clock in the Tx LPI mode

The following operations are performed when bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, the CSR clock can be gated off. If the MAC RX is not in LPI mode when the CSR clock is gated off, the events on the MAC receiver do not get reported or updated in the CSR. To restore the CSR clock, switch on the CSR clock when the MAC has exited TX LPI mode. After the CSR clock is resumed, reset bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* to exit the MAC from LPI mode.

57.9.12 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:

1. Program TRGTMODSEL[1:0] bit to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) as start time of PPS signal output.
2. Program the start time value in the Target Time registers (register *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
3. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)* Register.
4. Program PPSCMD[3:0] of *PPS control register (ETH_MACPPSCR)* to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the Target Time registers.

Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has elapsed. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be higher than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in *PPS width register (ETH_MACPPSWR)*.
3. Program PPSCMD[3:0] bits of *PPS control register (ETH_MACPPSCR)* to generate a single pulse after the previous pulse is deasserted. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in Target Time registers.

   If this command is given before the previous pulse becomes low, then the new command overwrites the previous command and the peripheral may generate only 1 extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:

1. Program TRGTMODSEL[1:0] bits to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS*

target time seconds register (ETH_MACPPSTTSR) and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) for start time of the PPS signal output.

2. Program the start time value in the Target Time registers (register *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
3. Program the interval value between the train of pulses on the PPS signal output in *PPS interval register (ETH_MACPPSIR)*.
4. Program the width of the PPS signal output in *PPS width register (ETH_MACPPSWR)*.
5. Program PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

   By default, the PPS pulse train is free-running unless it is stopped by issuing a 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands.
6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in *PPS target time nanoseconds register (ETH_MACPPSTTNR)* is reset before programming the Target Time registers again.
7. Program the PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

The pulse train can be stopped at any time by programming 0101 in the PPSCMD[3:0] field.

Similarly, the Stop Pulse train command (given in Step 7) can be canceled by programming PPSCMD[3:0] bits to 0110 before the time (programmed at step 6) has elapsed.

The pulse train generation can be stopped by programming PPSCMD[3:0] to 0011 before the start time programmed at step 2) has elapsed.

Generating an interrupt without affecting the PPS

TRGTMODSEL[1:0] bits in *PPS control register (ETH_MACPPSCR)* enable you to program the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL[1:0] bits of *PPS control register (ETH_MACPPSCR)* to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

   If TRGTMODSEL[1:0] bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

Note: The TSTRGTERR0 bit in Timestamp status register (ETH_MACTSSR) is set when the programmed target time is smaller (that is corresponds to a time in the past) compared to

the system time in the System time seconds register (ETH_MACSTSR) and System time nanoseconds register (ETH_MACSTNR).

An interrupt is generated (sbd_intr_o) if the TSIE bit in the Interrupt enable register (ETH_MACIER) is set.

Therefore, to avoid unwanted interrupt, the correct writing order is as follow:

1. *PPS target time nanoseconds register (ETH_MACPPSTTNR).*
2. *PPS target time seconds register (ETH_MACPPSTTSR).*
3. *PPS interval register (ETH_MACPPSIR).*
4. *PPS width register (ETH_MACPPSWR).*
5. *PPSCTRL[3:0] and PPSCTRL[3:0] and PPSEN0 bitfields of PPS control register (ETH_MACPPSCR).*

57.9.13 Programming guidelines for IEEE 1588 auxiliary snapshot

The IEEE 1588 auxiliary snapshot function is used to capture the auxiliary snapshot for a trigger event received on the ptp_aux_ts_trig_i input.

To program the IEEE 1588 auxiliary snapshot function, complete the following steps:

1. Program the TSIE bit in the *Interrupt enable register (ETH_MACIER)* to enable interrupt generation when the auxiliary snapshot is captured in the FIFO.
2. Program the ATSENx bit of the *Auxiliary control register (ETH_MACACR)* to enable the capture of the auxiliary snapshot based on the respective trigger input.
3. If the software does not want the captured auxiliary snapshot to be read, program the ATSFC bit of *Auxiliary control register (ETH_MACACR)* anytime to clear the auxiliary snapshot FIFO.
4. When TSIS is set or an interrupt is generated (due to TSIE programmed to 1), read the *Timestamp status register (ETH_MACTSSR)*, *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)* to access the captured auxiliary snapshot.

Note: When the auxiliary snapshot is captured in the FIFO for a trigger event, the following bitfields are updated:

- *AUXTSTRIG (status bit indicating that an auxiliary snapshot has been captured), ATSSTN (trigger event input source identifying the first available entry in the FIFO), and ATSNS (indicating the number of auxiliary snapshots available in the FIFO) bitfields of the Timestamp status register (ETH_MACTSSR).*
- *TSIS (indicating that an interesting event occurred in the IEEE 1588 timestamp function) bitfield in the Interrupt status register (ETH_DMAISR).*

The AUXTSTRIG bit is cleared on read (or set when the RCWE bit of the CSR software control register (ETH_MACCSRSWCR) is set).

Note: When the captured auxiliary snapshot is ignored because that the FIFO is full, the ATSSTM bit, which indicates that a trigger has been missed, is set.

57.9.14 Programming guidelines for TSO

The TCP Segmentation Offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet.

Follow the steps below to program TSO:

1. Program TSE bit of the corresponding *Channel transmit control register (ETH_DMACTXCR)* to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
   – MSS[13:0] of *Channel control register (ETH_DMACCR)*
   – or MSS in the context descriptor

   If MSS[13:0] field is programmed in both *Channel control register (ETH_DMACCR)* and in the context descriptor, the latest software programmed sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first descriptor. This buffer must not hold any payload bytes. The payload is allocated to Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.

57.9.15 Programming guidelines to perform VLAN filtering on the receiver

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program *VLAN tag register (ETH_MACVTR)* for the following bit to select the filtering method:
   – ETV: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   – VTHM: VLAN Tag Hash Table Match Enable.
   – ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   – ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   – DOVLTC: Ignores VLAN Type for Tag Match
   – VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL bit in *VLAN tag register (ETH_MACVTR)* for the 12-bit or 16-bit VLAN tag.
3. If VLAN tag Hash filtering is enabled, program *VLAN Hash table register (ETH_MACVHTR)*.
   – When the ETV bit is reset, the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the *VLAN Hash table register (ETH_MACVHTR)*.
   – When ETV bit is set, the upper 4 bits of the calculated CRC-32 of VLAN tag are used to index the content of *VLAN Hash table register (ETH_MACVHTR)*.

   For example, when ETV bit is set, a hash value of 0b1000 selects bit 8 of the VLAN Hash table. When ETV bit is reset, a hash value of 0b1000 selects bit 7 of the VLAN Hash table.

57.10 Descriptors

57.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- **Normal descriptors**

  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

- **Context descriptors**

  The context descriptors are used to provide control information applicable to the packet to be transmitted.

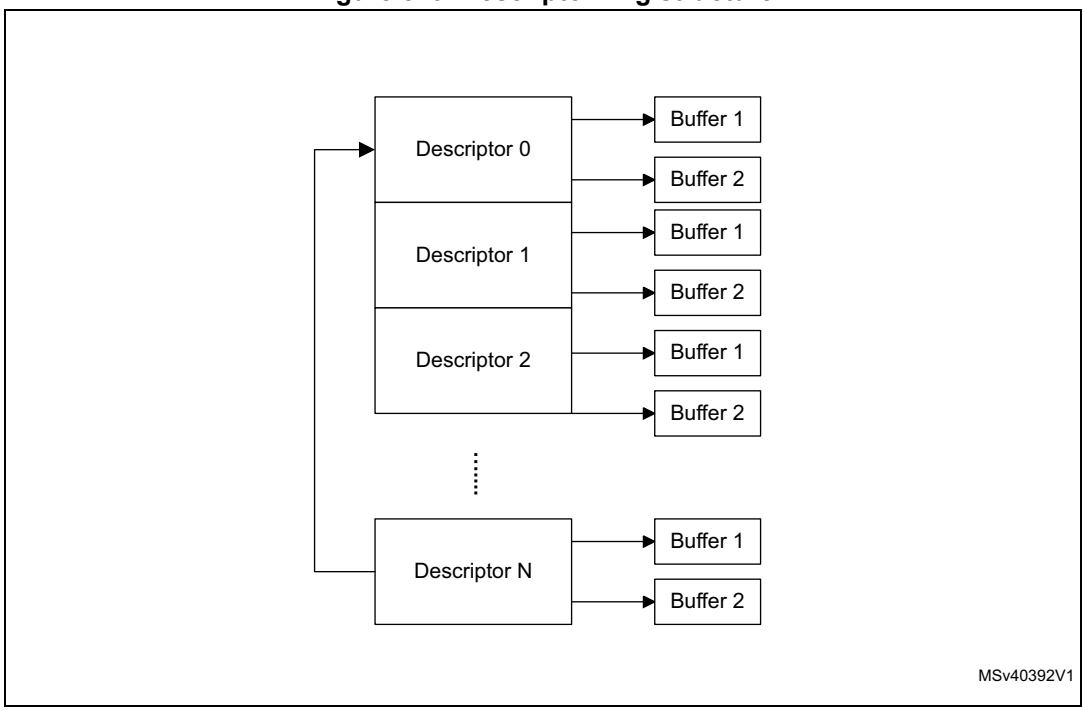
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.

57.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

Figure 829. Descriptor ring structure



In a ring structure, descriptors are separated by the 32-bit word number programmed in the DSL field of the *Channel control register (ETH_DMACCR)*). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- *Channel Tx descriptor ring length register (ETH_DMACTXRLR)*)
- *Channel Rx descriptor ring length register (ETH_DMACRXRLR)*)

The *Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)* or *Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)* contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the descriptor tail pointer register and update the tail pointer so that the following condition is met:

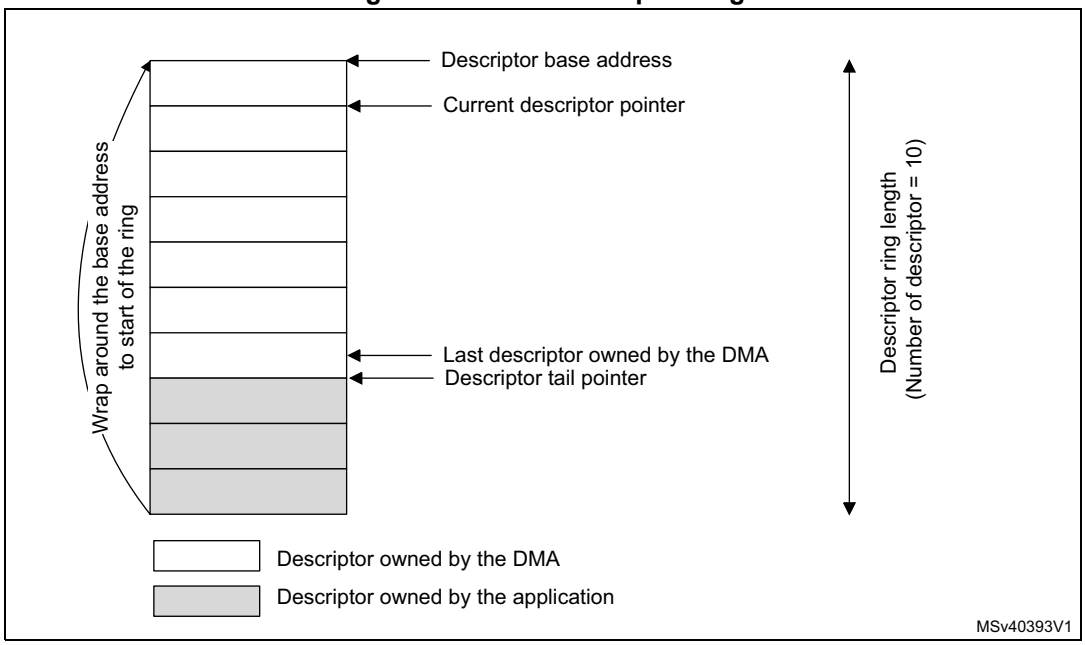
```
Descriptor Tail Pointer > Current Descriptor Pointer;
```

Note: When the application sets the descriptors into the end-of-descriptor ring, continues setting from the start of the descriptor list and updating the descriptor tail pointer, the above condition is not satisfied. The DMA detects such a case, processes the descriptors until the end of the ring and then continues processing from the start of the descriptor list until the current descriptor pointer is equal to the descriptor tail pointer.

When the application programs the descriptor tail pointer beyond the end of the descriptor ring, the DMA interprets this as the availability of one more complete ring of descriptors, and after processing the last descriptor in the ring continues to process descriptors from the start of the descriptor list. If this is not intended, the application must program the descriptor tail pointer to the start of the descriptor list after setting the last descriptor in the ring.

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 830: DMA descriptor ring*.

Figure 830. DMA descriptor ring



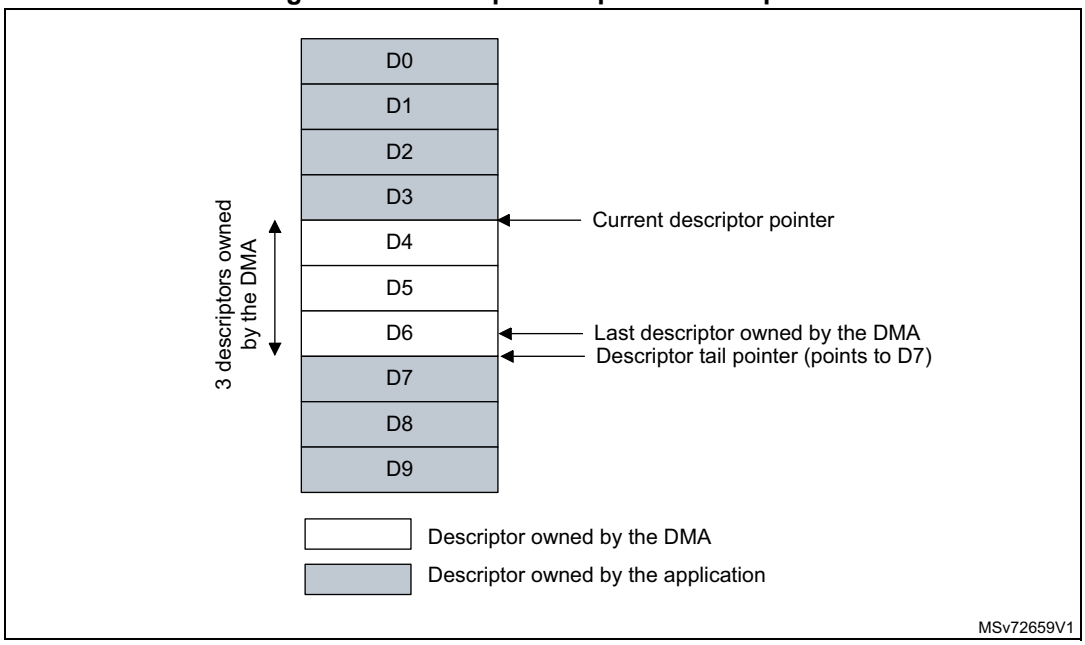
For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.

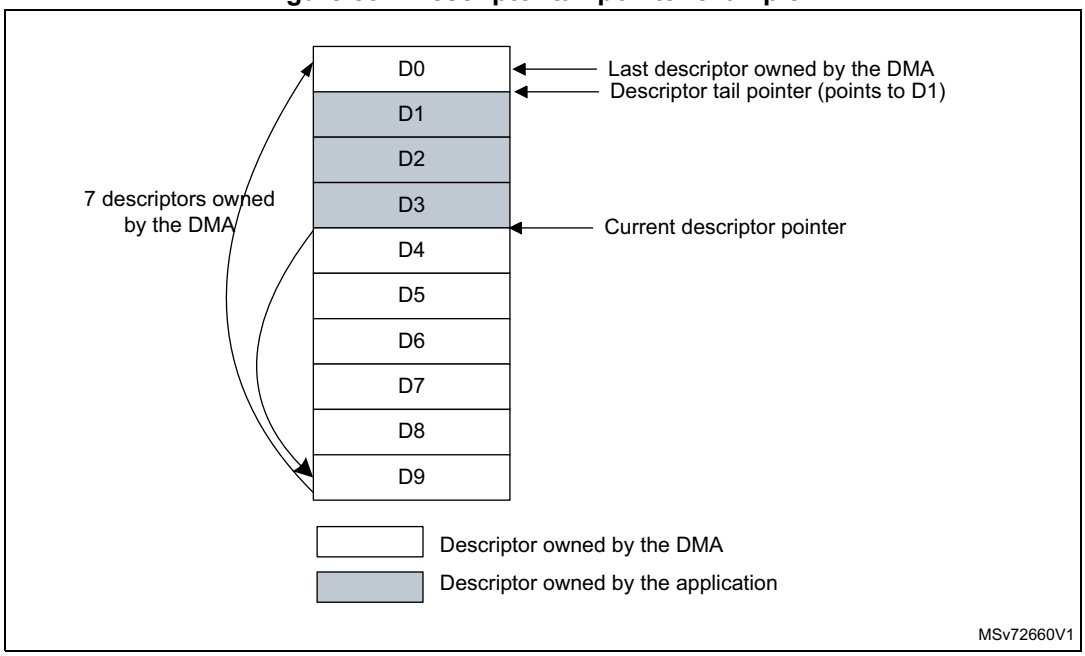
Descriptor tail pointer handling examples

Figure 831. Descriptor tail pointer example 1



When the current descriptor pointer points to descriptor 4 (D4) and the descriptor tail pointer points to descriptor 7 (D7), the last descriptor owned by the DMA is descriptor 6 (D6) and there are three descriptors (D4, D5, D6) available for the DMA, as shown in *Figure 831: Descriptor tail pointer example 1*.

Figure 832. Descriptor tail pointer example 2



As shown in *Figure 832: Descriptor tail pointer example 2*, the application frees four descriptors (D7, D8, D9 and D0) and updates the descriptor tail pointer to point to descriptor 1 (D1). The last descriptor owned by the DMA is descriptor 0 (D0). The descriptors referenced between the current descriptor pointer (D4) and the descriptor tail pointer (D1) are available to the DMA.

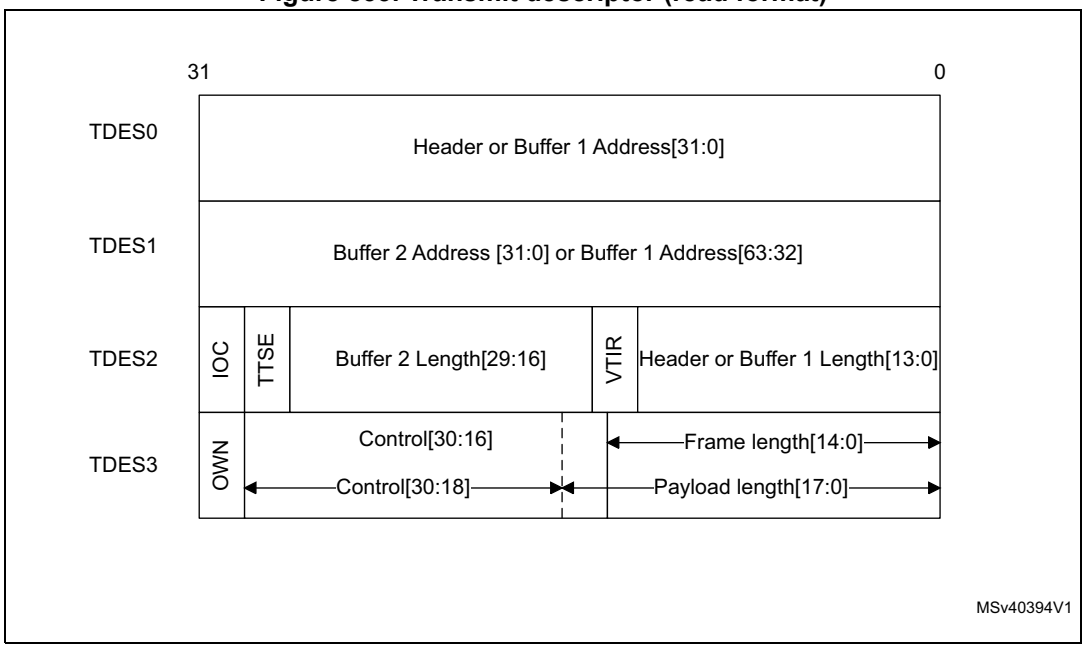
57.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format

Transmit normal descriptor (read format)

Figure 833 shows the Read format for Transmit normal descriptor. *Table 673* to *Table 676* provide a detailed description of all Transmit normal descriptors (read format).

Figure 833. Transmit descriptor (read format)



- TDES0 normal descriptor (read format)

Table 673. TDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer or TSO Header Address Pointer** These bits indicate either the physical address of Buffer 1 or the TSO Header Address pointer when the following bits are set: – TSE bit of TDES3 – FD bit of TDES3

- TDES1 normal descriptor (read format)

Table 674. TDES1 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 or Buffer 1 Address Pointer:** These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment.

- TDES2 normal descriptor (read format)

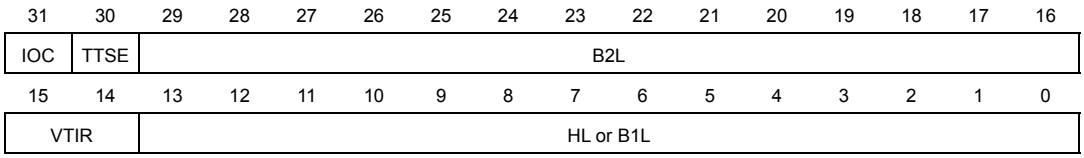


Table 675. TDES2 normal descriptor (read format)

Bits	Name	Description
31	IOC	**Interrupt on completion:** This bit sets the TI bit in the *Channel status register (ETH_DMACSR)*) when the present packet transmission is complete.
30	TTSE	**Transmit Timestamp Enable** This bit enables the IEEE1588 timestamping for Transmit packet referenced by the descriptor.
29:16	B2L	**Buffer 2 Length** The driver sets this field. When set, this field indicates Buffer 2 length.
15:14	VTIR	**VLAN Tag Insertion or Replacement:** These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows: 00: Do not add a VLAN tag. 01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 10: Insert a VLAN tag with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. 11: Replace the VLAN tag in packets with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. This option should be used only with the VLAN packets.

Table 675. TDES2 normal descriptor (read format) (continued)

Bits	Name	Description
13:0	HL or B1L	**Header length or buffer 1 length** For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length. If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes. If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length.

- TDES3 normal descriptor (read format)

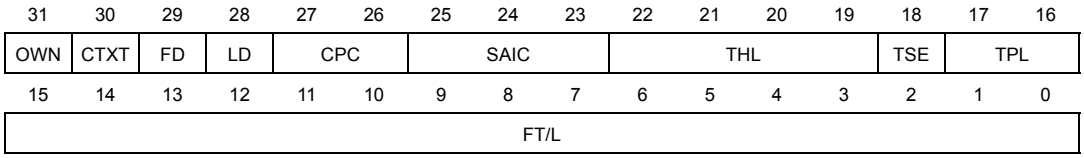


Table 676. TDES3 normal descriptor (read format)

Bits	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptor.
29	FD	**First Descriptor** When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value.

Table 676. TDES3 normal descriptor (read format) (continued)

Bits	Name	Description
27:26	CPC	**CRC Pad Control** This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following: 00: CRC and Pad Insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 01: CRC Insertion (Disable Pad Insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 10: Disable CRC Insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 11: CRC Replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.
25:23	SAIC	**SA Insertion Control** These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[24:23]: 00: Do not include the source address 01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. 11: Reserved These bits are valid when the First Segment control bit (TDES3 [29]) is set.
22:19	THL	**THL**: TCP Header Length If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. THL value must be equal to 2 for UDP header. This field is valid only for the first descriptor.
18	TSE	**TCP Segmentation Enable** When this bit is set, the DMA performs the TCP/UDP segmentation for a packet. This bit is valid only if the FD bit is set.

Table 676. TDES3 normal descriptor (read format) (continued)

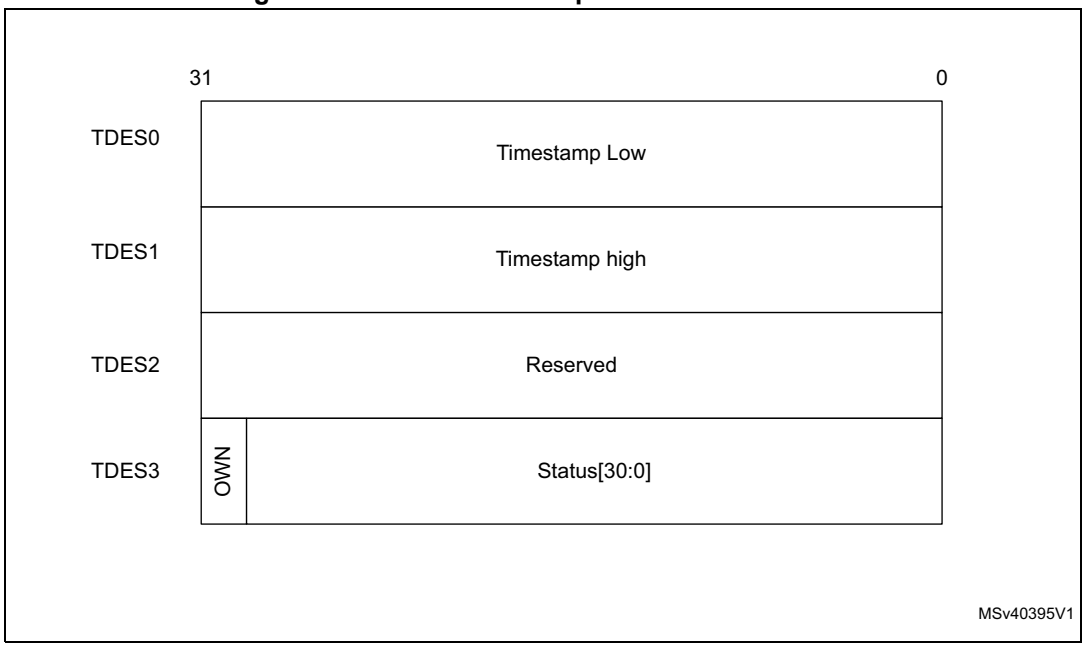
Bits	Name	Description
17:16	CIC/TPL	**Checksum Insertion Control or TCP Payload Length** These bits control the checksum calculation and insertion. They can take the following values: 00: Checksum insertion disabled. 01: Only IP header checksum calculation and insertion are enabled. 10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 11: IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the TSE bit is reset.When the TSE bit is set, it contains the upper bits [17:16] of the TCP Payload length. This allows the TCP packet length field to be spanned across TDES3[17:0] to provide 256 Kbyte packet length support.
15	TPL	**Reserved or TCP Payload Length** When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0].
14:0	FL/TPL	**Reserved or TCP Payload Length** When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length. This length does not include Ethernet header or TCP/UDP/IP header length. When the TSE bit is reset, this bit is reserved.

Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 834 shows the write-back format for Transmit normal descriptors. *Table 677* to *Table 680* provide a detailed description of all Transmit Normal descriptors (Write-Back Format).

Figure 834. Transmit descriptor write-back format



- TDES0 normal descriptor (write-back format)

Table 677. TDES0 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES1 normal descriptor (write-back format)

Table 678. TDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES2 normal descriptor (write-back format)

Table 679. TDES2 normal descriptor (write-back format)(1)

Bit	Description
31:0	Reserved

1. This format is only applicable to the last descriptor of a packet.

- TDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	Reserved										TTSS	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	PCE	LoC	NC	LC	EC	CC				ED	UF	DB	IHE

Table 680. TDES3 normal descriptor (write-back format)(1)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptors.
29	FD	**First Descriptor** This bit indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27:18	Reserved	
17	TTSS	**Tx Timestamp Status** This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set.
16	Reserved	

Table 680. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
15	ES	**Error Summary** This bit indicates the logical OR of the following bits: TDES3[0]: IP Header Error TDES3[14]: Jabber Timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error
14	JT	**Jabber Timeout** This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the *Operating mode configuration register (ETH_MACCR)* is not set.
13	FF	**Packet Flushed** This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	**Payload Checksum Error** This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.
11	LoC	**Loss of Carrier** This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.
10	NC	**No Carrier** This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	**Late Collision** This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode). This bit is not valid if Underflow Error is set.

Table 680. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
8	EC	**Excessive Collision** This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	**Collision Count** This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	**Excessive Deferral** This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times if DC bit is set in the *Operating mode configuration register (ETH_MACCR)*.
2	UF	**Underflow Error** This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty Transmit Buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register.
1	DB	**Deferred Bit** This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the Half-duplex mode.
0	IHE	**IP Header Error** When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.

1. This format is only applicable to the last descriptor of a packet.

Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, and VLAN Tag ID for VLAN insertion feature. Write-back is only done on a context descriptor to reset the OWN bit.

Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.

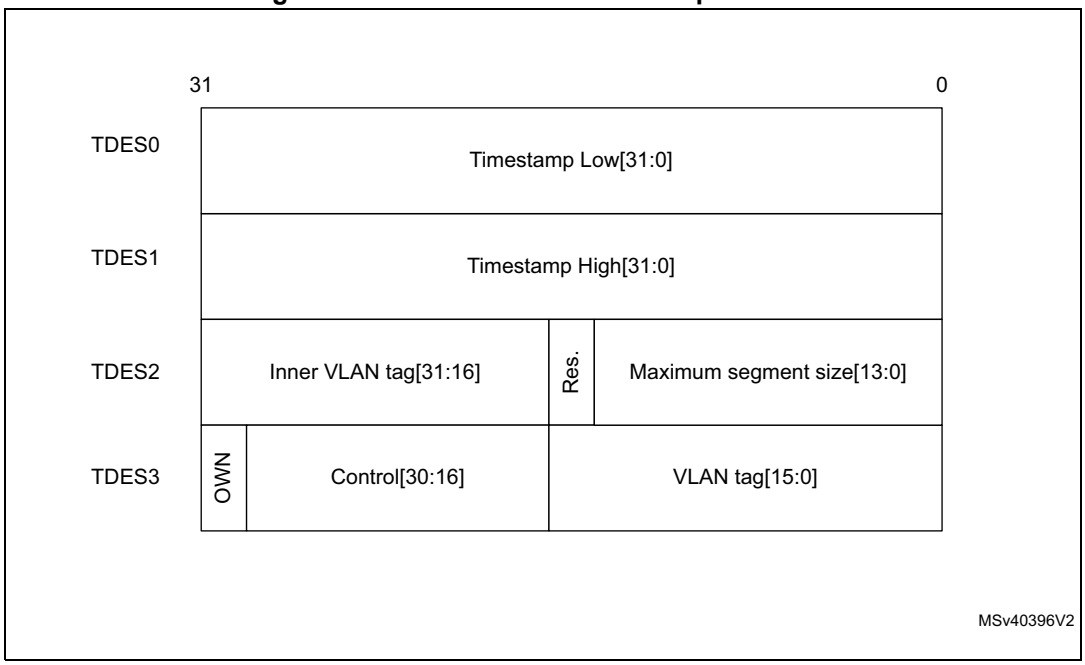
When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag

stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 835 shows the format for Transmit context descriptors. *Table 681* to *Table 684* provide a detailed description of all Transmit context descriptors.

Figure 835. Transmit context descriptor format



- TDES0 context descriptor (read format)

Table 681. TDES0 context descriptor

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)

Table 682. TDES1 context descriptor

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES2 context descriptor (read format)

Table 683. TDES2 context descriptor

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
15:14	Reserved	
13:0	MSS	**Maximum Segment Size** This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

- TDES3 context descriptor (read format)

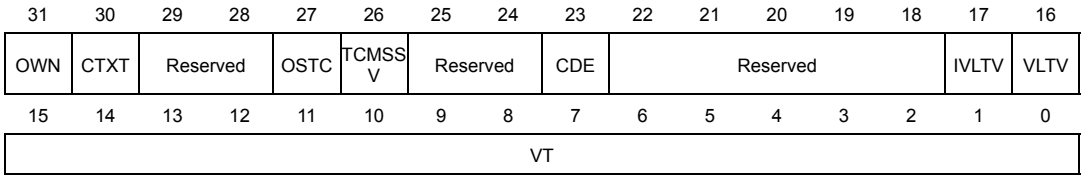


Table 684. TDES3 context descriptor

Bit	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit immediately after a read operation.
30	CTXT	**Context Type** This bit should be set to 1 for context descriptor.
29:28	Reserved	
27	OSTC	**One-Step Timestamp Correction Enable** When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.

Table 684. TDES3 context descriptor (continued)

Bit	Name	Description
26	TCMSSV	**One-Step Timestamp Correction Input or MSS Valid** When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	Reserved	
23	CDE	**Context Descriptor Error** When this bit is set, it indicates that the context descriptor is incorrect. The DMA sets this bit during write-back while closing the context descriptor. The Context Descriptor errors can be: – Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. – All 1s. – CD, LD, and FD bits set to 1. *Note: When a Context Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel status register (ETH_DMACSR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.* *This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported)*
22:20	Reserved	
19:18	IVTIR	Inner VLAN Tag Insert or Replace When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. This bitfield has the following values: 00: Do not add the inner VLAN tag. 01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. 10: Insert an inner VLAN tag with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. 11: Replace the inner VLAN tag in packets with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. This option should be used only with the VLAN frames.
17	IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.

Table 684. TDES3 context descriptor (continued)

Bit	Name	Description
16	VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the *VLAN inclusion register (ETH_MACVIR)* is reset.

57.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following Receive descriptors are present:

- Normal descriptors with read and write-back formats
- Context descriptors

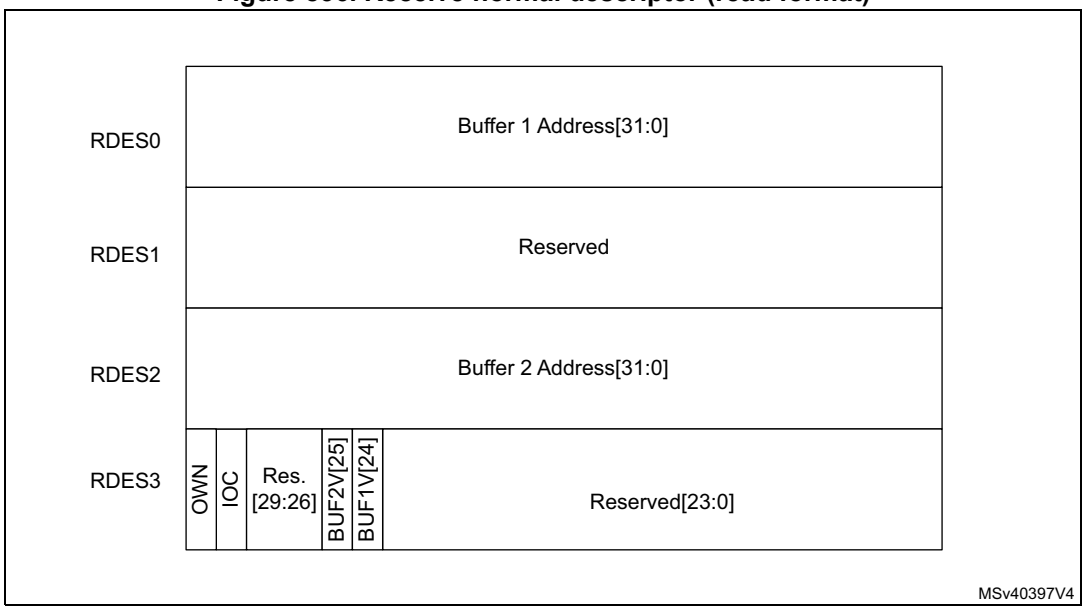
All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see *Figure 836: Receive normal descriptor (read format)* for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in *Figure 837: Receive normal descriptor (write-back format)*.

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA writes the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in *Figure 838: Receive context descriptor*.

Receive normal descriptor (read format)

Figure 836 shows the read format for Receive normal descriptors. *Table 685* to *Table 688* provide a detailed description of all Receive normal descriptors (read format).

Figure 836. Receive normal descriptor (read format)



Note: In the Receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.

- RDES0 normal descriptor (read format)

Table 685. RDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. The application can program a byte-aligned address for this buffer, which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES2[1:0]=0 in case of 64-/128-bit configuration) as zero. However, the packet data is shifted by the actual offset as given in the buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

- RDES1 normal descriptor (read format)

Table 686. RDES1 normal descriptor (read format)

Bit	Name	Description

- RDES2 normal descriptor (read format)

Table 687. RDES2 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 Address Pointer** These bits indicate Buffer 2 physical address. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores RDES2[1:0]=0 and writes to the complete location.

- RDES3 normal descriptor (read format)

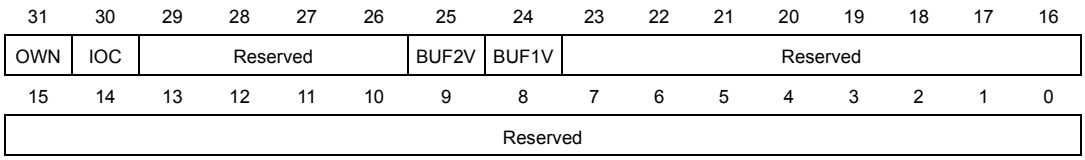


Table 688. RDES3 normal descriptor (read format)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full

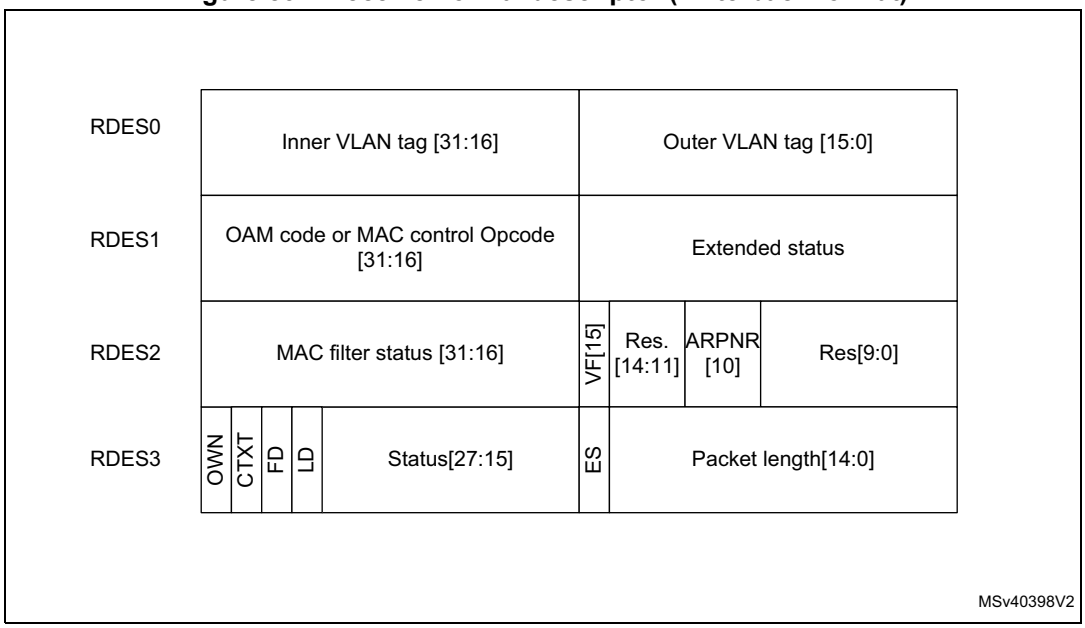
Table 688. RDES3 normal descriptor (read format)

30	IOC	**Interrupt Enabled on Completion** When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Reserved	
25	BUF2V	**Buffer 2 Address Valid** When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.
24	BUF1V	**Buffer 1 Address Valid** When set, this indicates to the DMA that the buffer 1 address specified in RDES0 is valid. The application must set this value if the address to which Buffer 1 address points in RDES0, can be used by the DMA to write received packet data.
23:0	Reserved	

Receive normal descriptor (write-back format)

Figure 837 shows the write-back format for Receive normal descriptors. *Table 689* to *Table 692* provide a detailed description of all Receive normal descriptors (write-back format).

Figure 837. Receive normal descriptor (write-back format)



- RDES0 normal descriptor (write-back format)

Table 689. RDES0 normal descriptor (write-back format)

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
15:0	OVT	**Outer VLAN Tag** This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

- RDES1 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPC															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD	TSA	PV	PFT	PMT				IPCE	IPCB	IPV6	IPV4	IPHE	PT		

Table 690. RDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:16	OPC	**OAM Subtype Code, or MAC Control Packet opcode** OAM Subtype Code If bits[18:16] of RDES3 are set to 111, this field contains the OAM subtype and code fields. MAC Control Packet opcode If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
15	TD	**Timestamp Dropped** This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.
14	TSA	**Timestamp Available** When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	**PTP Version** 1: Received PTP message in IEEE 1588 version 2 format 0: Received PTP message in IEEE 1588 version 1 format
12	PFT	**PTP Packet Type** This bit indicates that the PTP message is sent directly over Ethernet.

Table 690. RDES1 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
11:8	PMT	**PTP Message Type** These bits are encoded to give the type of the message received: 0000: No PTP message received 0001: SYNC (all clock types) 0010: Follow_Up (all clock types) 0011: Delay_Req (all clock types) 0100: Delay_Resp (all clock types) 0101: Pdelay_Req (in peer-to-peer transparent clock) 0110: Pdelay_Resp (in peer-to-peer transparent clock) 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000: Announce 1001: Management 1010: Signaling 1011–1110: Reserved 1111: PTP packet with Reserved message type
7	IPCE	**IP Payload Error** When this bit is set, it indicates either of the following: – The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. – The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. – The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.
6	IPCB	**IP Checksum Bypassed** This bit indicates that the checksum offload engine is bypassed.
5	IPV6	**IPv6 header Present** This bit indicates that an IPV6 header is detected.
4	IPV4	**IPV4 Header Present** This bit indicates that an IPV4 header is detected.
3	IPHE	**IP Header Error** – When this bit is set, it indicates either of the following: – The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. – The IP datagram version is not consistent with the Ethernet Type value. – Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.

Table 690. RDES1 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
2:0	PT	**Payload Type** These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE): 000: Unknown type or IP/AV payload not processed 001: UDP 010: TCP 011: ICMP 100: IGMP if IPV4 Header Present bit is set Others: reserved. If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

- RDES2 normal descriptor (write-back format)

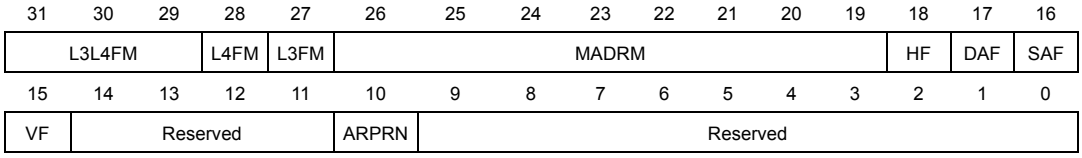


Table 691. RDES2 normal descriptor (write-back format)

Bit	Name	Description
31:29	L3L4FM	**Layer 3 and Layer 4 Filter Number Matched** These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: – 000: Filter 0 – 001: Filter 1 – 010: Filter 2 – 011: Filter 3 – 100: Filter 4 – 101: Filter 5 – 110: Filter 6 – 111: Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter.
28	L4FM	**Layer 4 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: – Layer 3 fields are not enabled and all enabled Layer 4 fields match – All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29].

Table 691. RDES2 normal descriptor (write-back format) (continued)

Bit	Name	Description
27	L3FM	**Layer 3 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: – All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed – All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29].
26:19	MADRM	**MAC Address Match or Hash Value** When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register.
18	HF	**Hash Filter Status** When this bit is set, it indicates that the packet passed the MAC address Hash filter. its[26:19] indicate the Hash value.
17	DAF	**Destination Address Filter Fail** When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
16	SAF	**SA Address Filter Fail** When this bit is set, it indicates that the packet failed the SA Filter in the MAC.
15	VF	**VLAN Filter Status** When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter.
14:11	Reserved	
10	ARPNR	**ARP Reply Not Generated** When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
9:0	Reserved	

- RDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	PL														

Table 692. RDES3 normal descriptor (write-back format)

Bit	Name	Description
31	OWN	**Own bit** 1: The DMA owns the descriptor. 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} possible values are: 00: Intermediate Descriptor 01: First Descriptor 10: Reserved 11: Descriptor error (due to all 1s) *Note: When a Descriptor error occurs, the Receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel status register (ETH_DMACSR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.*
29	FD	**First Descriptor** When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. Refer to the CTXT bit description for details on how to use the CTXT bit and FD bit together.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	**Receive Status RDES2 Valid** When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	**Receive Status RDES1 Valid** When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	**Receive Status RDES0 Valid** When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.

Table 692. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
24	CE	**CRC Error** When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	**Giant Packet** When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	**Receive Watchdog Timeout** When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	**Overflow Error** When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
20	RE	**Receive Error** When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception.
19	DE	**Dribble Bit Error** When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18:16	LT	**Length/Type Field** This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 000: The packet is a length packet 001: The packet is a type packet. 011: The packet is a ARP Request packet type 100: The packet is a type packet with VLAN Tag 101: The packet is a type packet with Double VLAN tag 110: The packet is a MAC Control packet type 111: The packet is a OAM packet type 010: Reserved

Table 692. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
15	ES	**Error Summary** When this bit is set, it indicates the logical OR of the following bits: RDES3[19]: Dribble Error RDES3[20]: Receive Error RDES3[21]: Overflow Error RDES3[22]: Watchdog Timeout RDES3[23]: Giant Packet RDES3[24]: CRC Error This field is valid only when the LD bit of RDES3 is set.
14:0	PL	**Packet Length** These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet.

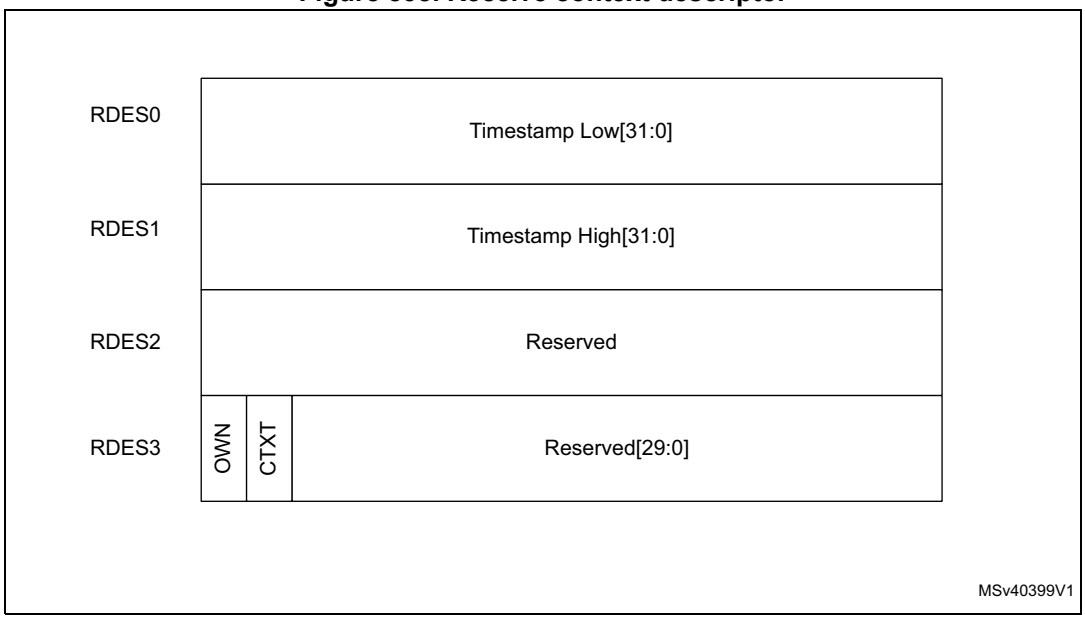
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

Figure 838 shows the format for Receive context descriptors. *Table 693* to *Table 696* provide a detailed description of all Receive context descriptors.

Figure 838. Receive context descriptor



- RDES0 context descriptor

Table 693. RDES0 context descriptor

Bit	Name	Description
31:0	RTSL	**Receive Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

- RDES1 context descriptor

Table 694. RDES1 context descriptor

Bit	Field	Description
31:0	RTSH	**Receive Packet Timestamp High** The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

- RDES2 context descriptor

Table 695. RDES2 context descriptor

Bit	Description
31:0	Reserved

- RDES3 context descriptor

Table 696. RDES3 context descriptor

Bit	Name	Description
31	OWN	**Own Bit** 1: The DMA owns the descriptor 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor.
29:0	Reserved	

57.11 Ethernet registers

57.11.1 Ethernet register maps

This section provides the following register maps:

- DMA registers (see *Section 57.11.2: Ethernet DMA registers*)
- MTL registers (see *Section 57.11.3: Ethernet MTL registers*)
- MAC registers including MMC registers (see *Section 57.11.4: Ethernet MAC and MMC registers*)

57.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000

Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

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

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

Bits 17:16 **INTM[1:0]**: Interrupt Mode

This field defines the interrupt mode of the Ethernet peripheral.
The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMACSR register changes depending on the INTM value (refer to *Table 672: Transfer complete interrupt behavior*).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PR[2:0]**: Priority ratio

These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set.
000: The priority ratio is 1:1
001: The priority ratio is 2:1
010: The priority ratio is 3:1
011: The priority ratio is 4:1
100: The priority ratio is 5:1
101: The priority ratio is 6:1
110: The priority ratio is 7:1
111: The priority ratio is 8:1

Bit 11 **TXPR**: Transmit priority

When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.

Bits 10:2 Reserved, must be kept at reset value.

Bit 1 **DA**: DMA Tx or Rx Arbitration Scheme

This bit specifies the arbitration scheme between the Transmit and Receive paths of all channels:

0: Weighted Round-Robin with Rx:Tx or Tx:Rx

The priority between the paths is according to the priority specified in Bits[14:12] and the priority weight is specified in the TXPR bit.

1: Fixed priority

The Tx path has priority over the Rx path when the TXPR bit is set. Otherwise, the Rx path has priority over the Tx path.

Bit 0 **SWR**: Software Reset

When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all clock domains. Before reprogramming any register, a value of zero should be read in this bit.

Note: The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.

System bus mode register (ETH_DMASBMR)

Address offset: 0x1004

Reset value: 0x0000 0000

The System bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RB**: Rebuild INCRx Burst

When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.

Bit 14 **MB**: Mixed Burst

When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).

Bit 13 Reserved, must be kept at reset value.

Bit 12 **AAL**: Address-Aligned Beats

When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **FB**: Fixed Burst Length

When this bit is set to 1, the AHB master initiates burst transfers of specified length (INCRx or SINGLE).

When this bit is set to 0, the AHB master initiates transfers of unspecified length (INCR) or SINGLE transfers.

Interrupt status register (ETH_DMAISR)

Address offset: 0x1008

Reset value: 0x0000 0000

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC0IS
															r

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

Bit 17 **MACIS**: MAC Interrupt Status

This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

Bit 16 **MTLIS**: MTL Interrupt Status

This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **DC0IS**: DMA Channel Interrupt Status

This bit indicates an interrupt event in DMA Channel. To reset this bit to 0, the software must read the corresponding register in DMA Channel to get the exact cause of the interrupt and clear its source.

Debug status register (ETH_DMADSR)

Address offset: 0x100C

Reset value: 0x0000 0000

The Debug status register gives the Receive and Transmit process status for DMA Channel for debugging purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXWHSTS
r	r	r	r	r	r	r	r								r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **TPS0[3:0]**: DMA Channel Transmit Process State
This field indicates the Tx DMA FSM state for Channel:
000: Stopped (Reset or Stop Transmit Command issued)
001: Running (Fetching Tx Transfer Descriptor)
010: Running (Waiting for status)
011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
100: Timestamp write state
101: Reserved for future use
110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
111: Running (Closing Tx Descriptor)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPS0[3:0]**: DMA Channel Receive Process State
This field indicates the Rx DMA FSM state for Channel:
000: Stopped (Reset or Stop Receive Command issued)
001: Running (Fetching Rx Transfer Descriptor)
010: Reserved for future use
011: Running (Waiting for Rx packet)
100: Suspended (Rx Descriptor Unavailable)
101: Running (Closing the Rx Descriptor)
110: Timestamp write state
111: Running (Transferring the received packet data from the Rx buffer to the system memory)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **AXWHSTS**: AHB Master Write Channel
When high, this bit indicates that the write channel of the AHB master FMSs are in non-idle state.

Channel control register (ETH_DMACCR)

Address offset: 0x1100

Reset value: 0x0000 0000

The DMA Channel control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8
											rw	rw	rw		rw

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:18 **DSL[2:0]**: Descriptor Skip Length

This bit specifies the 32-bit word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.

When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PBLX8**: 8xPBL mode

When this bit is set, the PBL value programmed in Bits[21:16] in *Channel transmit control register (ETH_DMACTXCR)* and in Bits[21:16] in *Channel receive control register (ETH_DMACRXCR)* is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

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

Bits 13:0 **MSS[13:0]**: Maximum Segment Size

This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of *Channel transmit control register (ETH_DMACTXCR)* is set.

The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

Channel transmit control register (ETH_DMACTXCR)

Address offset: 0x1104

Reset value: 0x0000 0000

The DMA Channel Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

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

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

Bits 21:16 **TXPBL[5:0]**: Transmit Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in ETH_DMACCR.

b) Set the TXPBL[5:0].

Note: The maximum value of TXPBL must be less than or equal to half the Tx Queue size (TQS field of *Tx queue operating mode register (ETH_MTLTXQOMR)*) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. The total locations in Tx Queue of size 2048 bytes is 512, TXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TSE**: TCP Segmentation Enabled

When this bit is set, the DMA performs the TCP segmentation for packets in Channel x. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **OSF**: Operate on Second Packet

When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ST**: Start or Stop Transmission Command

When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the base address of the Transmit list set by the ETH_DMACTXDLAR register.

– The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMACSR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMACTXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMACTXDLAR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

Channel receive control register (ETH_DMACRXCR)

Address offset: 0x1108

Reset value: 0x0000 0000

The DMA Channel Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

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

Bit 31 **RPF**: DMA Rx Channel Packet Flush

When this bit is set to 1, the Ethernet peripheral automatically flushes the packet from the Rx queues destined to DMA Rx Channel when the DMA Rx Channel is stopped after a system bus error has occurred. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing happens on the Read side of the Rx queue.

When this bit is set to 0 the Ethernet peripheral does not flush the packet in the Rx queue destined to DMA Rx Channel after the DMA is stopped due to a system bus error.

This might cause head-of-line blocking in the corresponding RxQueue.

Note: The stopping of packet flow from a Rx DMA Channel to the application by setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA channels. In Dynamic mapping mode, setting RPF bit in ETH_DMACRXCR register might flush packets from unintended Rx Queues which are destined to the stopped Rx DMA Channel.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:16 **RXPBL[5:0]**: Receive Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in the ETH_DMACCR.

b) Set the RXPBL[5:0].

Note: The maximum value of RXPBL must be less than or equal to half the Rx Queue size (RQS field of Rx queue operating mode register (ETH_MTLRXQOMR)) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the MTL Rx Queue Controller is transferring data to MAC.The total locations in Rx Queue of size 2048 bytes is 512, RXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 Reserved, must be kept at reset value.

Bits 14:1 **RBSZ[13:0]**: Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes.

Note: The buffer size must be a multiple of 4. This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 4, it may result into an undefined behavior.

The LSB bits (1:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0 **SR**: Start or Stop Receive

When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the address set by the *Channel Rx descriptor list address register (ETH_DMACRXDLAR)*.

– The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMACSR is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the *Channel Rx descriptor list address register (ETH_DMACRXDLAR)*, the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

Channel Tx descriptor list address register (ETH_DMACTXDLAR)

Address offset: 0x1114

Reset value: 0x0000 0000

Channel Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Writing to this register is permitted only when the Tx DMA has stopped, that is, the ST bit is cleared in ETH_DMACTXCR register. When stopped, this register can be written with a new descriptor list address. When the ST bit is set, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is cleared, the DMA takes the descriptor address where it was stopped earlier.

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

Bits 31:0 **TDESLA[31:0]**: Start of Transmit List

This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel Rx descriptor list address register (ETH_DMACRXDLAR)

Address offset: 0x111C

Reset value: 0x0000 0000

The Channel Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in ETH_DMACRXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

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

Bits 31:0 **RDESLA[31:0]**: Start of Receive List

This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)

Address offset: 0x1120

Reset value: 0x0000 0000

The Channel Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

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

Bits 31:0 **TDT[31:0]**: Transmit Descriptor Tail Pointer

This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)

Address offset: 0x1128

Reset value: 0x0000 0000

The Channel Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

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

Bits 31:0 **RDT[31:0]**: Receive Descriptor Tail Pointer

This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

Channel Tx descriptor ring length register (ETH_DMACTXRLR)

Address offset: 0x112C

Reset value: 0x0000 0000

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.

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

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

Bits 9:0 **TDRL[9:0]**: Transmit Descriptor Ring Length

This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. It is recommended to put a minimum ring descriptor length of 4.

For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

Channel Rx descriptor ring length register (ETH_DMACRXRLR)

Address offset: 0x1130

Reset value: 0x0000 0000

The Channel Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **ARBS[7:0]**: Alternate Receive Buffer Size

Indicates size in bytes for Buffer 1 when ARBS[7:0] is programmed to a non-zero value. When ARBS[7:0] = 0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ[13:0] field of *Channel receive control register (ETH_DMACRXCR)*.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **RDRL[9:0]**: Receive Descriptor Ring Length

This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors.

For example, you can program any value up to 0x3FF in this field. This field is 10-bit wide. If you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

Channel interrupt enable register (ETH_DMACIER)

Address offset: 0x1134

Reset value: 0x0000 0000

The Channel Interrupt Enable register enables the interrupts reported by the Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NIE**: Normal Interrupt Summary Enable

When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the *Channel status register (ETH_DMACSR)*:
Bit 0: Transmit interrupt
Bit 2: Transmit buffer unavailable
Bit 6: Receive interrupt
Bit 11: Early Receive interrupt
When this bit is reset, the normal interrupt summary is disabled.

Bit 14 **AIE**: Abnormal Interrupt Summary Enable

When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the *Channel status register (ETH_DMACSR)*:
Bit 1: Transmit Process Stopped
Bit 7: Rx Buffer Unavailable
Bit 8: Receive Process Stopped
Bit 9: Receive Watchdog Timeout
Bit 10: Early transmit interrupt
Bit 12: Fatal Bus Error
When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 **CDEE**: Context Descriptor Error Enable

When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 **FBEE**: Fatal Bus Error Enable

When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 **ERIE**: Early receive interrupt Enable

When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled.

Bit 10 **ETIE**: Early Transmit Interrupt Enable

When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 **RWTE**: Receive Watchdog Timeout Enable

When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled.

Bit 8 **RSE**: Receive Stopped Enable

When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.

Bit 7 **RBUE**: Receive Buffer Unavailable Enable

When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.

Bit 6 **RIE**: Receive Interrupt Enable

When this bit is set along with the NIE bit, the receive interrupt is enabled. When this bit is reset, the receive interrupt is disabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBUE**: Transmit Buffer Unavailable Enable

When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

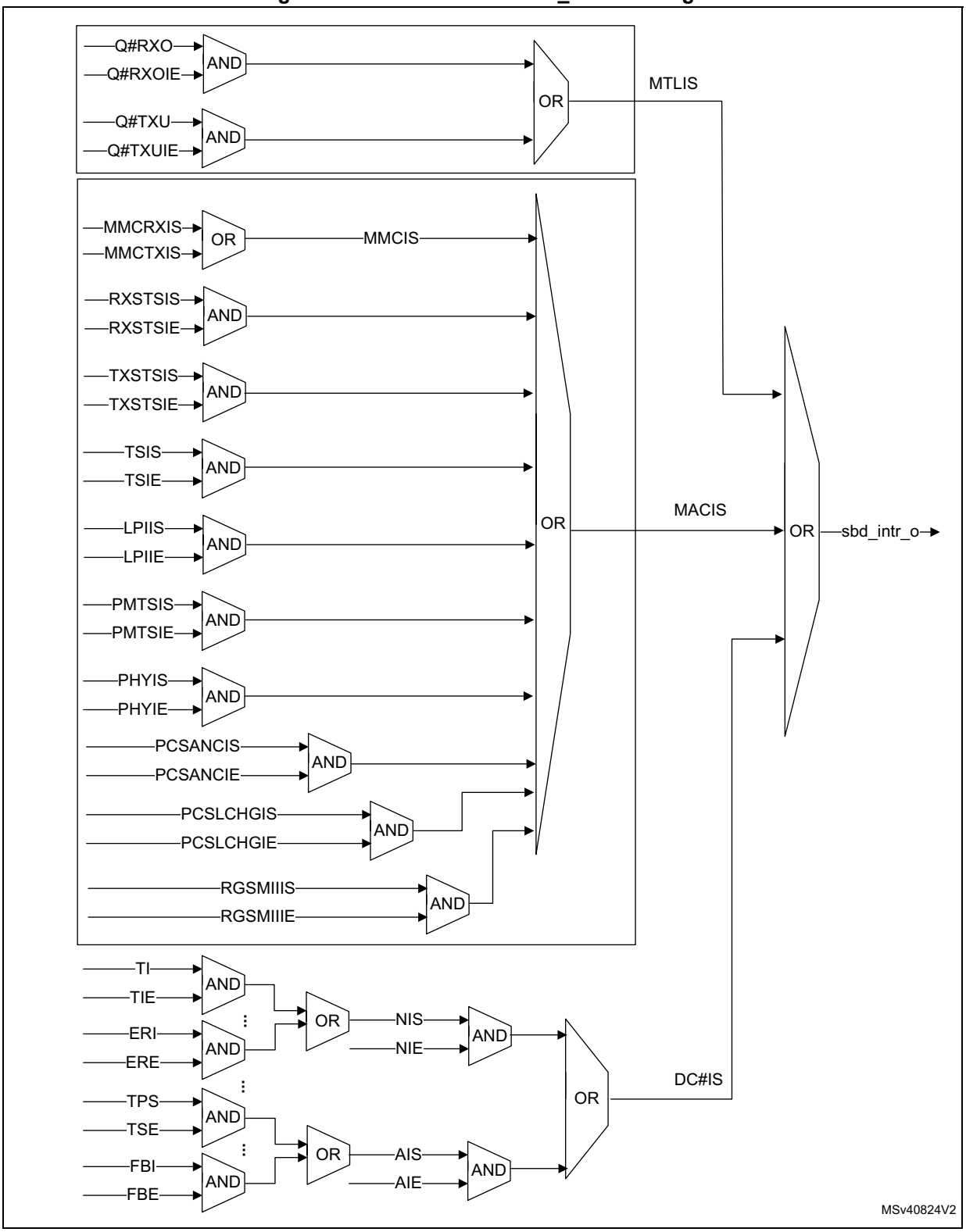
Bit 1 **TXSE**: Transmit Stopped Enable

When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0 **TIE**: Transmit Interrupt Enable

When this bit is set along with the NIE bit, the transmit interrupt is enabled. When this bit is reset, the transmit interrupt is disabled.

Figure 839. Generation of ETH_DMAISR flags



Channel Rx interrupt watchdog timer register (ETH_DMACRXIWTR)

Address offset: 0x1138

Reset value: 0x0000 0000

The receive interrupt Watchdog Timer register indicates the watchdog timeout for receive interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the *Channel status register (ETH_DMACSR)*.

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

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

Bits 17:16 **RWTU[1:0]**: Receive Interrupt Watchdog Timer Count Units

This bitfield indicates the number of system clock cycles corresponding to one unit in RWT[7:0] bitfield.
00: 256
01: 512
10: 1024
11: 2048
For example, when RWT[7:0] = 2 and RWTU[1:0] = 1, the watchdog timer is set for 2 × 512 = 1024 system clock cycles.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RWT[7:0]**: Receive Interrupt Watchdog Timer Count

This bitfield indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.
The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMACSR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].
When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

Channel current application transmit descriptor register (ETH_DMACCATXDR)

Address offset: 0x1144

Reset value: 0x0000 0000

The Channel Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

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

Bits 31:0 **CURTDESAPTR[31:0]**: Application Transmit Descriptor Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel current application receive descriptor register (ETH_DMACCARXDR)

Address offset: 0x114C

Reset value: 0x0000 0000

The Channel Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

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

Bits 31:0 **CURRDESAPTR[31:0]**: Application Receive Descriptor Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel current application transmit buffer register (ETH_DMACCATXBR)

Address offset: 0x1154

Reset value: 0x0000 0000

The Channel Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

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

Bits 31:0 **CURTBUFAPTR[31:0]**: Application Transmit Buffer Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel current application receive buffer register (ETH_DMACCARXBR)

Address offset: 0x115C

Reset value: 0x0000 0000

The Channel Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

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

Bits 31:0 **CURRBUFAPTR[31:0]**: Application Receive Buffer Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel status register (ETH_DMACSR)

Address offset: 0x1160

Reset value: 0x0000 0000

The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]		
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rw	rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1

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

Bits 21:19 **REB[2:0]**: Rx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AHB interface.
Bit [2]: Error during data transfer by Rx DMA when 1, no rror during data transfer by Rx DMA when 0.
Bit[1]: Error during descriptor access when 1, error during data buffer access when 0
Bit[0]: Error during read transfer when 1, error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bits 18:16 **TEB[2:0]**: Tx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AHB interface.
Bit[2]: Error during data transfer by Tx DMA when 1, no error during data transfer by Tx DMA when 0
Bit[1]: Error during descriptor access when 1, error during data buffer access when 0
Bit[0]: Error during read transfer when 1, Error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15 **NIS**: Normal Interrupt Summary

Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
Bit 0: Transmit interrupt
Bit 2: Transmit buffer unavailable
Bit 6: Receive interrupt
Bit 11: Early receive interrupt
Only unmasked bits (interrupts for which interrupt enable is set in ETH_DMACIER register) affect the Normal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14 **AIS**: Abnormal Interrupt Summary

Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
Bit 1: Transmit Process Stopped
Bit 7: Receive Buffer Unavailable
Bit 8: Receive Process Stopped
Bit 10: Early transmit interrupt
Bit 12: Fatal Bus Error
Bit 13: Context Descriptor Error
Only unmasked bits affect the Abnormal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13 **CDE**: Context Descriptor Error

This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one‘s descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.

Bit 12 **FBE**: Fatal Bus Error

This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.

Bit 11 **ERI**: Early Receive Interrupt
This bit indicates that the DMA filled the first data buffer of the packet.The RI bit of this register automatically clears this bit.

Bit 10 **ETI**: Early Transmit Interrupt
This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO.

Bit 9 **RWT**: Receive Watchdog Timeout
This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.

Bit 8 **RPS**: Receive Process Stopped
This bit is asserted when the Rx process enters the Stopped state.

Bit 7 **RBU**: Receive Buffer Unavailable
This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor.

Bit 6 **RI**: Receive Interrupt
This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.
The reception remains in the Running state.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBU**: Transmit Buffer Unavailable
This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSi field of the *Debug status register (ETH_DMADSR)* register explains the Transmit Process state transitions.
To resume processing the Transmit descriptors, the application should do the following:
1. Change the ownership of the descriptor by setting Bit 31 of TDES3.
2. Issue a Transmit Poll Demand command.
For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel.

Bit 1 **TPS**: Transmit Process Stopped
This bit is set when the transmission is stopped.

Bit 0 **TI**: Transmit Interrupt
This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

Channel missed frame count register (ETH_DMACMFCR)

Address offset: 0x116C

Reset value: 0x0000 0000

This register has the number of packet counter that got dropped by the DMA either due to bus error or due to programing RPF field in *Channel receive control register (ETH_DMACRXCR)* register.

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

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MFCO**: Overflow status of the MFC Counter

When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MFC[10:0]**: Dropped Packet Counters

This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in *Channel receive control register (ETH_DMACRXCR)*. The counter gets cleared when this register is read.

Ethernet DMA register map and reset values

Table 697. ETH_DMA common register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000	ETH_DMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]		Res.	PR[2:0]			TXPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.			DA	SWR
	Reset value															0	0		0	0	0	0										0	0
0x1004	ETH_DMASBMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RB	MB	Res.	AAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FB
	Reset value																	0	0		0												0
0x1008	ETH_DMAISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC0IS
	Reset value															0	0																0
0x100C	ETH_DMADSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	AXWHSTS
	Reset value																	0	0	0	0	0	0	0	0								0

Table 698. ETH_DMA_CH register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1100	ETH_DMACCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1104	ETH_DMACTXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPBL[5:0]						Res.	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	Res.			ST
	Reset value											0	0	0	0	0	0				0								0				0
0x1108	ETH_DMACRXCR	RPF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPBL[5:0]						Res.	RBSZ[13:0]											Res.	Res.	Res.	SR
	Reset value	0										0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0				0
0x110C - 0x1110	Reserved																																
0x1114	ETH_DMACTXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1118	Reserved																																
0x111C	ETH_DMACRXDLAR	RDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1120	ETH_DMACTXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 698. ETH_DMA_CH register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1124	Reserved																																
0x1128	ETH_DMACRXDTPR	RDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x112C	ETH_DMACTXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1130	ETH_DMACRXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value									0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x1134	ETH_DMACIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1138	ETH_DMACRXIWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x113C-0x1140	Reserved																																
0x1144	ETH_DMACCATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1148	Reserved																																
0x0114C	ETH_DMACCARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1150	Reserved																																
0x1154	ETH_DMACCATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1158	Reserved																																
0x115C	ETH_DMACCARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1160	ETH_DMACSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1164 - 0x1168	Reserved																																

Table 698. ETH_DMA_CH register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x116C	ETH_DMACMFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

57.11.3 Ethernet MTL registers

Operating mode register (ETH_MTLOMR)

Address offset: 0x0C00

Reset value: 0x0000 0000

The Operating Mode register establishes the Transmit and Receive operating modes and commands.

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

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

Bit 9 **CNTCLR**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRST bit, CNTPRST has precedence.

Bit 8 **CNTPRST**: Counters Preset

When this bit is set:

- *Tx queue underflow register (ETH_MTLTXQUR)* is initialized/preset to 0x7F0.
- Missed Packet and Overflow Packet counters in *Rx queue missed packet and overflow counter register (ETH_MTLRXQMPOCR)* is initialized/preset to 0x7F0

This bit is cleared automatically.

Bit 7 Reserved, must be kept at reset value.

Bits 6:2 Reserved, must be kept at reset value.

Bit 1 **DTXSTS**: Drop Transmit Status

When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

Bit 0 Reserved, must be kept at reset value.

Interrupt status register (ETH_MTLISR)

Address offset: 0x0C20

Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q0IS
															r

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

Bit 0 **Q0IS**: Queue interrupt status

This bit indicates that an interrupt has been generated by Queue. To reset this bit, read ETH_MTLQICSR register to identify the interrupt cause and clear the source.

Tx queue operating mode register (ETH_MTLTXQOMR)

Address offset: 0x0D00

Reset value: 0x0007 0008

The Queue Transmit Operating Mode register establishes the Transmit queue operating modes and commands.

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **TQS[2:0]**: Transmit queue size

This field indicates the size of the allocated transmit queues in blocks of 256 bytes. Queue size range from 256 bytes (TQS=0b000) to 2048 bytes (TQS=0b111).

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

Bits 6:4 **TTC[2:0]**: Transmit Threshold Control

These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.

000: 32
001: 64
010: 96
011: 128
100: 192
101: 256
110: 384
111: 512

Bits 3:2 **TXQEN[1:0]**: Transmit Queue Enable

This field is used to enable/disable the transmit queue .

00: Not enabled
10: Enabled
Others: Reserved, must not be used.

Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.

Bit 1 **TSF**: Transmit Store and Forward

When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped.

Bit 0 **FTQ**: Flush Transmit Queue

When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the ETH_MTLTXQOMR register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (eth_tx_clk) should be active.

Tx queue underflow register (ETH_MTLTXQUR)

Address offset: 0x0D04

Reset value: 0x0000 0000

The Queue Underflow Counter register contains the counter for packets aborted because of Transmit queue underflow and packets missed because of Receive queue packet flush

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

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **UFCNTOVF**: Overflow Bit for Underflow Packet Counter

This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened.

Bits 10:0 **UFFRMCNT[10:0]**: Underflow Packet Counter

This field indicates the number of packets aborted by the controller because of Tx queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read.

Tx queue debug register (ETH_MTLTXQDR)

Address offset: 0x0D08

Reset value: 0x0000 0000

The Queue Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

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

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

Bits 22:20 **STXSTSF[2:0]**: Number of Status Words in Tx Status FIFO of Queue

This field indicates the current number of status in the Tx Status FIFO of this queue.
When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PTXQ[2:0]**: Number of Packets in the Transmit Queue

This field indicates the current number of packets in the Tx queue.
When the DTXSTS bit of *Operating mode register (ETH_MTLOMR)* register is set to 1, this field does not reflect the number of packets in the Transmit queue.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TXSTSFSTS**: MTL Tx Status FIFO Full Status

When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.

Bit 4 **TXQSTS**: MTL Tx Queue Not Empty Status

When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission.

Bit 3 **TWCSTS**: MTL Tx Queue Write Controller Status
When high, this bit indicates that the MTL Tx queue Write Controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx Queue Read Controller Status
This field indicates the state of the Tx Queue Read Controller:
00: Idle state
01: Read state (transferring data to the MAC transmitter)
10: Waiting for pending Tx Status from the MAC transmitter
11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit Queue in Pause
When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the Full-duplex only mode) because of the following:
– Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
– Reception of 802.3x Pause packet when PFC is disabled

Queue interrupt control status register (ETH_MTLQICSR)

Address offset: 0x0D2C

Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS
							rw								rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUNFIS
							rw								rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive Queue Overflow Interrupt Enable
When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

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

Bit 16 **RXOVFIS**: Receive Queue Overflow Interrupt Status
This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TXUIE**: Transmit Queue Underflow Interrupt Enable

When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **TXUNFIS**: Transmit Queue Underflow Interrupt Status

This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.

Rx queue operating mode register (ETH_MTLRXQOMR)

Address offset: 0x0D30

Reset value: 0x0070 0000

The Queue Receive operating Mode register establishes the Receive queue operating modes and command.

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

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

Bits 22:20 **RQS[2:0]**: Receive Queue Size

This field is read-only and the configured Rx FIFO size in blocks of 256 bytes is reflected in the reset value. The size of the Queue is (RQS + 1) × 256 bytes.

Bits 19:7 Reserved, must be kept at reset value.

Bit 6 **DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets

When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.

When this bit is reset, all error packets are dropped if the FEP bit is reset.

Bit 5 **RSF**: Receive Queue Store and Forward

When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Bit 4 **FEP**: Forward Error Packets

When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped.

When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.

Bit 3 **FUP**: Forward Undersized Good Packets

When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTC[1:0]**: Receive Queue Threshold Control

These bits control the threshold level of the MTL Rx queue (in bytes):

00: 64

01: 32

10: 96

11: 128

The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred.

This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

Rx queue missed packet and overflow counter register (ETH_MTLRXQMPOCR)

Address offset: 0x0D34

Reset value: 0x0000 0000

The Queue missed packet and overflow counter registers contain the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

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

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTOVF**: Missed Packet Counter Overflow Bit

When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed Packet Counter

This field indicates the number of packets missed by the Ethernet peripheral because the application requested to flush the packets for this queue. This counter is reset when this register is read.

This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

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

Bit 11 **OVFCNTOVF**: Overflow Counter Overflow Bit

When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow Packet Counter

This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.

Rx queue debug register (ETH_MTLRXQDR)

Address offset: 0x0D38

Reset value: 0x0000 0000

The Queue Receive Debug register gives the debug status of various blocks related to the Receive queue.

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PRXQ[13:0]**: Number of Packets in Receive Queue

This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256 Kbyte/16 bytes = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **RXQSTS[1:0]**: MTL Rx Queue Fill-Level Status

This field gives the status of the fill-level of the Rx queue:
00: Rx queue empty
01: Rx queue fill-level below flow-control deactivate threshold
10: Rx queue fill-level above flow-control activate threshold
11: Rx queue full

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **RRCSTS[1:0]**: MTL Rx Queue Read Controller State

This field gives the state of the Rx queue Read controller:
00: Idle state
01: Reading packet data
10: Reading packet status (or timestamp)
11: Flushing the packet data and status

Bit 0 **RWCSTS**: MTL Rx Queue Write Controller Active Status

When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.

Ethernet MTL register map and reset values

Table 699. ETH_MTL register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C00	ETH_MTLOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTCLR	CNTPRST	Res.	Res.	Res.	Res.	Res.	Res.	DTXSTS	Res.
	Reset value																							0	0							0	
0x0C04 - 0x0C1C	Reserved																																
0x0C20	ETH_MTLISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q0IS
	Reset value																																0
0x0C24 - 0x0CFC	Reserved																																
0x0C40	Reserved																																
0x0D00	ETH_ MTLTXQOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value														1	1	1										0	0	0	1	0	0	0
0x0D04	ETH_ MTLTXQUR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D08	ETH_ MTLTXQDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D0C - 0x0D28	Reserved																																
0x0D2C	ETH_MTLQICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXUNFIS
	Reset value								0								0								0								0
0x0D30	ETH_ MTLRXQOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value										1	1	1														0	0	0	0		0	0

Table 699. ETH_MTL register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D34	ETH_ MTLRXQMPOCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0D38	ETH_ MTLRXQDR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res	Res	Res.	Res.	Res.	Res		RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0		0	0	0
0xD3C-0xD58	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

57.11.4 Ethernet MAC and MMC registers

Operating mode configuration register (ETH_MACCR)

Address offset: 0x0000

Reset value: 0x0000 8000

The MAC Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	Res.	JD	JE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **ARPEN**: ARP Offload Enable

When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus.

When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.

Bits 30:28 **SARC[2:0]**: Source Address Insertion or Replacement Control

This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:

010:the MAC inserts the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

011: the MAC replaces the content of the MAC Address 0 registers (*MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

110: the MAC inserts the content of the MAC Address 1 registers (*MAC Address x high register (ETH_MACAxHR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets

111: the MAC replaces the content of the MAC Address 1 registers (*MAC Address x high register (ETH_MACAxHR)* and *MAC Address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

Others: Reserved, must not be used.

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bit 27 **IPC**: Checksum Offload

When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.

The Layer 3 and Layer 4 Packet Filter feature automatically selects the IPC Full Checksum Offload Engine on the Receive side. When this feature is enabled, you must set the IPC bit.

Bits 26:24 **IPG[2:0]**: Inter-Packet Gap

These bits control the minimum IPG between packets during transmission.

000: 96 bit times

001: 88 bit times

010: 80 bit times

...

111: 40 bit times

This range of minimum IPG is valid in Full-duplex mode.

In the Half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.

When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.

The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23 **GPSLCE**: Giant Packet Size Limit Control Enable

When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.

When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).

The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22 **S2KP**: IEEE 802.3as Support for 2K Packets

When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.

When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 700: Giant Packet Status based on S2KP and JE Bits*.

Note: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21 **CST**: CRC stripping for Type packets

When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.

Note: For information about how the settings of the ACS bit and this bit impact the packet length, see Table 701: Packet Length based on the CST and ACS bits.

Bit 20 **ACS**: Automatic Pad or CRC Stripping

When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.

When this bit is reset, the MAC passes all incoming packets to the application, without any modification.

Note: For information about how the settings of CST bit and this bit impact the packet length, see Table 701: Packet Length based on the CST and ACS bits.

Bit 19 **WD**: Watchdog Disable

When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.

When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **JD**: Jabber Disable

When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.

When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.

Bit 16 **JE**: Jumbo Packet Enable

When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.

For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 700: Giant Packet Status based on S2KP and JE Bits*.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **FES**: MAC Speed

This bit selects the speed in the 10/100 Mbps mode:

0: 10 Mbps

1: 100 Mbps

Bit 13 **DM**: Duplex Mode

When this bit is set, the MAC operates in the Full-duplex mode in which it can transmit and receive simultaneously.

Bit 12 **LM**: Loopback Mode

When this bit is set, the MAC operates in the loopback mode at MII. The MII Rx clock input (eth_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Bit 11 **ECRSFD**: Enable Carrier Sense Before Transmission in Full-duplex mode

When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the Full-duplex mode. The MAC starts the transmission only when the CRS signal is low.

When this bit is reset, the MAC transmitter ignores the status of the CRS signal.

Bit 10 **DO**: Disable Receive Own

When this bit is set, the MAC disables the reception of packets when the ETH_TX_EN is asserted in the Half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.

This bit is not applicable in the Full-duplex mode. This bit is reserved and read-only (RO) with default value in the Full-duplex-only configurations.

Bit 9 **DCRS**: Disable Carrier Sense During Transmission

When this bit is set, the MAC transmitter ignores the MII CRS signal during packet transmission in the Half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.

When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

Bit 8 **DR**: Disable Retry

When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.

When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the Half-duplex mode.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **BL[1:0]**: Back-Off Limit

The back-off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision:

00: k= min (n, 10)

01: k = min (n, 8)

10: k = min (n, 4)

11: k = min (n, 1)

where n = retransmission attempt

The random integer r takes the value in the range $0 <= r < 2^k$.

This bit is applicable only in the Half-duplex mode.

Bit 4 **DC**: Deferral Check

When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode.

Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII.

The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.

When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.

This bit is applicable only in the Half-duplex mode.

Bits 3:2 **PRELEN[1:0]**: Preamble Length for Transmit packets

These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the Full-duplex mode.

00: 7 bytes of preamble

01: 5 bytes of preamble

10: 3 bytes of preamble

11: Reserved, must not be used

Bit 1 **TE**: Transmitter Enable

When this bit is set, the Tx state machine of the MAC is enabled for transmission on the MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 **RE**: Receiver Enable

When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the MII interface.

Table 700 shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

Table 700. Giant Packet Status based on S2KP and JE Bits(1)

Length/Type Field	Received Packet Length	S2KP	JE	Giant Packet Status
Untagged packet	> 1,518	0	0	1
	> 2,000	1	0	1
	> 9,018	x	1	1
VLAN tagged packet	> 1,522	0	0	1
	> 2,000	1	0	1
	> 9,022	x	1	1

1. For all other combinations, the Giant Packet status is 0.

Table 701 shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

Table 701. Packet Length based on the CST and ACS bits

Received Packet Length	CST	ACS	FCS Stripping Done
< 1,536	x	0	No
	x	1	Yes (for Ethernet packets)
≥ 1,536	0	x	No
	1	x	Yes (for Type packets)

Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004

Reset value: 0x0000 0000

The MAC Extended Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC
		rw	rw	rw	rw	rw	rw						rw	rw	rw

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **EIPG[4:0]**: Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits) along with IPG field in *Operating mode configuration register (ETH_MACCR)*, gives the minimum IPG greater than 96 bit times in steps of 8 bit times. For example:

- EIPG = 0 and IPG = 0 give 104 bit times
- EIPG = 0 and IPG = 1 give 112 bit times
- EIPG = 0 and IPG = 2 give 120 bit times
- ..
- EIPG = 7 and IPG = 31 give 2144 bit times

Bit 24 **EIPGEN**: Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in *Operating mode configuration register (ETH_MACCR)* together as minimum IPG greater than 96 bit times in steps of 8 bit times.

When this bit is reset, the MAC ignores EIPG field and interprets IPG field in *Operating mode configuration register (ETH_MACCR)* as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-duplex mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **USP**: Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the *MAC Address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC Address x low register (ETH_MACAxLR)*. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).

When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.

Bit 17 **SPEN**: Slow Protocol Detection Enable

When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid subtypes.
When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16 **DCRCC**: Disable CRC Checking for Received Packets

When this bit is set, the MAC receiver does not check the CRC field in the received packets.
When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

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

Bits 13:0 **GPSL[13:0]**: Giant Packet Size Limit

If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.
For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. For double VLAN tagged packets, the MAC adds 8 bytes to the programmed value. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

Packet filtering control register (ETH_MACPFR)

Address offset: 0x0008

Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE
rw										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RA**: Receive All

When this bit is set, the MAC receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word.
When this bit is reset, the receiver module passes only those packets to the application that pass the SA or DA address filter.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DNTU**: Drop Non-TCP/UDP over IP Packets

When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.

Bit 20 **IPFE**: Layer 3 and Layer 4 Filter Enable

When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.

When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **VTFE**: VLAN Tag Filter Enable

When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HPF**: Hash or Perfect Filter

When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.

When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Bit 9 **SAF**: Source Address Filter Enable

When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.

When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

Note: According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8 **SAIF**: SA Inverse Filtering

When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.

When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6 **PCF[1:0]**: Pass Control Packets

These bits control the forwarding of all control packets (including unicast and multicast Pause packets).

00: The MAC filters all control packets from reaching the application.

01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.

10: The MAC forwards all control packets to the application even if they fail the Address filter.

11: The MAC forwards the control packets that pass the Address filter.

Bit 5 **DBF**: Disable Broadcast Packets

When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.

When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4 **PM**: Pass All Multicast

When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.

Bit 3 **DAIF**: DA Inverse Filtering

When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast

When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.

When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 1 **HUC**: Hash Unicast

When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.

When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 0 **PR**: Promiscuous Mode

When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

Watchdog timeout register (ETH_MACWTR)

Address offset: 0x000C

Reset value: 0x0000 0000

The Watchdog Timeout register controls the watchdog timeout for received packets.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PWE**: Programmable Watchdog Enable

When this bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in *Operating mode configuration register (ETH_MACCR)* register.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WTO[3:0]**: Watchdog Timeout

When the PWE bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes

0x1: 3 Kbytes

0x2: 4 Kbytes

0x3: 5 Kbytes

..

0xC: 14 Kbytes

0xD: 15 Kbytes

0xE: 16383 Bytes

0xF: Reserved, must not be used

Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

Hash Table 0 register (ETH_MACHT0R)

Address offset: 0x0010

Reset value: 0x0000 0000

The Hash Table Register 0 contains the first lower 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 0b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT31T0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT31T0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT31T0[31:0]**: MAC Hash Table First 32 Bits
This field contains the first 32 Bits [31:0] of the Hash table.

Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014

Reset value: 0x0000 0000

The Hash Table 1 register contains the upper 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT63T32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT63T32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT63T32[31:0]**: MAC Hash Table Second 32 Bits
This field contains the second 32 Bits [63:32] of the Hash table.

VLAN tag register (ETH_MACVTR)

Address offset: 0x0050

Reset value: 0x0000 0000

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV
rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

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

Bit 31 **EIVLRXS**: Enable Inner VLAN Tag in Rx Status

When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **EIVLS[1:0]**: Enable Inner VLAN Tag Stripping on Receive

This field indicates the stripping operation on inner VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 27 **ERIVLT**: Enable Inner VLAN Tag

When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.The ERSVLM bit and DOVLTC bit determines which VLAN type is enabled for filtering.

Bit 26 **EDVLP**: Enable Double VLAN Processing

When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).

Bit 25 **VTHM**: VLAN Tag Hash Table Match Enable

When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHTR register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table.
When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison.
When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison.
When this bit is reset, the VLAN Hash Match operation is not performed.

Bit 24 **EVLRXS**: Enable VLAN Tag in Rx status

When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EVLS[1:0]**: Enable VLAN Tag Stripping on Receive

This field indicates the stripping operation on the outer VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 20 **DOVLTC**: Disable VLAN Type Check

When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19 **ERSVLM**: Enable Receive S-VLAN Match

When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18 **ESVL**: Enable S-VLAN

When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17 **VTIM**: VLAN Tag Inverse Match Enable

When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16 **ETV**: Enable 12-Bit VLAN Tag Comparison

When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:0 **VL[15:0]**: VLAN Tag Identifier for Receive Packets

This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag
When the ETV bit is set, only the VID is used for comparison.
If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.

VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058

Reset value: 0x0000 0000

When the VTHM bit of *VLAN tag register (ETH_MACVTR)* register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of *VLAN tag register (ETH_MACVTR)* register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

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

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VLHT[15:0]**: VLAN Hash Table
This field contains the 16-bit VLAN Hash Table.

VLAN inclusion register (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

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

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.

10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Inner VLAN inclusion register (ETH_MACIVIR)

Address offset: 0x0064

Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

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

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion

The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.

10: VLAN tag insertion

The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement

The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.
The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Tx Queue flow control register (ETH_MACQTXFCR)

Address offset: 0x0070

Reset value: 0x0000 0000

The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

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

Bits 31:16 **PT[15:0]**: Pause Time

This field holds the value to be used in the Pause Time field in the Tx control packet. I

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DZPQ**: Disable Zero-Quanta Pause

When this bit is set, it disables the automatic generation of the zero-quanta Pause packets.
When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Bits 6:4 **PLT[2:0]**: Pause Low Threshold

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

000: Pause Time minus 4 Slot Times (PT -4 slot times)

001: Pause Time minus 28 Slot Times (PT -28 slot times)

010: Pause Time minus 36 Slot Times (PT -36 slot times)

011: Pause Time minus 144 Slot Times (PT -144 slot times)

100: Pause Time minus 256 Slot Times (PT -256 slot times)

101: Pause Time minus 512 Slot Times (PT -512 slot times)

110 to 111: Reserved, must not be used

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface.

This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TFE**: Transmit Flow Control Enable

Full-duplex mode: when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

Half-duplex mode: when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0 **FCB_BPA**: Flow Control Busy or Backpressure Activate

This bit initiates a Pause packet in the Full-duplex mode and activates the backpressure function in the Half-duplex mode if the TFE bit is set.

Full-Duplex mode: this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

Half-duplex mode: When this bit is set (and TFE bit is set) in the Half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the Full-duplex mode, the BPA is automatically disabled.

Rx flow control register (ETH_MACRXFCR)

Address offset: 0x0090

Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

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

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

Bit 1 **UP**: Unicast Pause Packet Detect

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in *MAC Address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC Address x low register (ETH_MACAxLR)*.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01 80 C2 00 00 01) is as specified in IEEE 802.1 Qbb-2011.

Bit 0 **RFE**: Receive Flow Control Enable

When this bit is set and the MAC is operating in Full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in Half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.

Interrupt status register (ETH_MACISR)

Address offset: 0x00B0

Reset value: 0x0000 0000

The Interrupt Status register contains the status of interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	Res.
	rc_r	rc_r	rc_r		r	r	r			r	r	r			

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIS**: Receive Status Interrupt

This bit indicates the status of received packets. This bit is set when the RWT bit is set in the *Rx Tx status register (ETH_MACRXTXSR)*. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 13 **TXSTSIS**: Transmit Status Interrupt

This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the *Rx Tx status register (ETH_MACRXTXSR)*:

- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber Timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 12 **TSIS**: Timestamp Interrupt Status

If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:

– The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.

– There is an overflow in the Seconds register.

– The Target Time Error occurred, that is, programmed target time already elapsed.

If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the *Timestamp status register (ETH_MACTSSR)*.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC Transmit Interrupt Status

This bit is set high when an interrupt is generated in the *MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 9 **MMCRXIS**: MMC Receive Interrupt Status

This bit is set high when an interrupt is generated in the *MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 8 **MMCIS**: MMC Interrupt Status

This bit is set high when MMCTXIS or MMCRXIS is set high. This bit is cleared only when all these bits are low.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **LPIIS**: LPI Interrupt Status

This bit is set for any LPI state entry or exit in the MAC transmitter or receiver. This bit is cleared when the TLPIEN bit of *LPI control and status register (ETH_MACLCSR)* is read.

Bit 4 **PMTIS**: PMT Interrupt Status

This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the *PMT control status register (ETH_MACPCSR)*.

Bit 3 **PHYIS**: PHY Interrupt

This bit is set when rising edge is detected on the ETH_PHY_INTN input. This bit is cleared when this register is read.

Bits 2:0 Reserved, must be kept at reset value.

Interrupt enable register (ETH_MACIER)

Address offset: 0x00B4

Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIE**: Receive Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 13 **TXSTSIE**: Transmit Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 12 **TSIE**: Timestamp Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **LPIIE**: LPI Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 4 **PMTIE**: PMT Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 3 **PHYIE**: PHY Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 2:0 Reserved, must be kept at reset value.

Rx Tx status register (ETH_MACRXTXSR)

Address offset: 0x00B8

Reset value: 0x0000 0000

The Receive Transmit Status register contains the Receive and Transmit Error status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
							rc_r			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RWT**: Receive Watchdog Timeout

This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **EXCOL**: Excessive Collisions

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after the first collision and the packet transmission is aborted.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 4 **LCOL**: Late Collision

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode.

This bit is not valid if the Underflow error occurs.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 3 **EXDEF**: Excessive Deferral

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)* and the DC bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 when Jumbo packet is enabled).

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **LCARR**: Loss of Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the loss of carrier occurred during packet transmission, that is, the ETH_CRS signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **NCARR**: No Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TJT**: Transmit Jabber Timeout

This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the *Operating mode configuration register (ETH_MACCR)*.

Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

PMT control status register (ETH_MACPCSR)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw			r	r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
					rw	rw			r	rc_r			rw	rw	rw

Bit 31 **RWKFILTRST**: Remote Wake-up Packet Filter Register Pointer Reset

When this bit is set, the remote wake-up packet filter register pointer is reset to 0. It is automatically cleared after 1 clock cycle.

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:24 **RWKPTR[4:0]**: Remote wake-up FIFO Pointer

This field gives the current value (0 to 7) of the Remote wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wake-up Packet Filter Register are transferred to the eth_rx_clk domain when a Write occurs to that register.

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **RWKPFE**: Remote wake-up Packet Forwarding Enable

When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet.

The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high.

Note: If Magic Packet Enable and wake-up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.

Bit 9 **GLBLUCAST**: Global Unicast

When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RWKPRCVD**: Remote wake-up Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read.

Bit 5 **MGKPRCVD**: Magic Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RWKPKTEN**: Remote wake-up Packet Enable

When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.

Bit 1 **MGKPKTEN**: Magic Packet Enable

When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0 **PWRDWN**: Power Down

When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wake-up Packet Enable bit is set high.

Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

Remote wake-up packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4

Reset value: 0x0000 0000

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

Bits 31:0 **MACRWKPFR[31:0]**: Remote wake-up packet filter

Refer to *Table 669*, *Table 670* and *Table 671* for details on register content and programming sequence.

The ETH_MACRWKPFR register at address 0x00C4 loads the wake-up Packet Filter register.

To load values in a wake-up Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

LPI control and status register (ETH_MACLCSR)

Address offset: 0x00D0

Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	Res.	PLS	LPIEN
										rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
						r	r					r	r	r	r

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

Bit 21 **LPITCSE**: LPI Tx Clock Stop Enable

When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.
If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.

Bit 20 **LPITE**: LPI Timer Enable

This bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPIEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.
After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again.
When LPITE is 0, the LPI Auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.

Bit 19 **LPITXA**: LPI Tx Automate

This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side.
If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTxQOMR, when the MAC is in the LPI mode, it exits the LPI mode.
When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PLS**: PHY Link Status

This bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.
When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16 **LPIEN**: LPI Enable

When this bit is set, it instructs the MAC transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.
This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **RLPIST**: Receive LPI State

When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.

Bit 8 **TLPIST**: Transmit LPI State

When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **RLPIEX**: Receive LPI Exit

When this bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 2 **RLPIEN**: Receive LPI Entry

When this bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 1 **TLPIEX**: Transmit LPI Exit

When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TLPIEN**: Transmit LPI Entry

When this bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

LPI timers control register (ETH_MACLTCR)

Address offset: 0x00D4

Reset value: 0x03E8 0000

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

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

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

Bits 25:16 **LST[9:0]**: LPI LS Timer

This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 sec) as defined in the IEEE standard.

Bits 15:0 **TWT[15:0]**: LPI TW Timer

This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

LPI entry timer register (ETH_MACLETR)

Address offset: 0x00D8

Reset value: 0x0000 0000

This register controls the Tx LPI entry timer. This counter is enabled only when LPITE bit of *LPI control and status register (ETH_MACLCSR)* register is set to 1.

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **LPIET[19:0]**: LPI Entry Timer

This field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1.

Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

One-microsecond-tick counter register (ETH_MAC1USTCR)

Address offset: 0x00DC

Reset value: 0x0000 0063

This register controls the generation of the Reference time (one-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 TIC_1US_CNTR[11:0]: 1 µs tick Counter

The application must program this counter so that the number of clock cycles of CSR clock is 1 µs (subtract 1 from the value before programming).
For example if the CSR clock is 100 MHz then this field needs to be programmed to 100 - 1 = 99 (which is 0x63).
This is required to generate the 1 µs events that are used to update some of the EEE related counters.

Version register (ETH_MACVR)

Address offset: 0x0110

Reset value: 0x0000 3242

The version register identifies the version of the Ethernet peripheral.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USERVER[7:0]								SNPSVER[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **USERVER[7:0]**: ST-defined version

Bits 7:0 **SNPSVER[7:0]**: IP version

Debug register (ETH_MACDR)

Address offset: 0x0114

Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS
													r	r	r

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **TFCSTS[1:0]**: MAC Transmit Packet Controller Status

This field indicates the state of the MAC Transmit Packet Controller module:
00: Idle state
01: Waiting for one of the following:
– Status of the previous packet
– IPG or backoff period to be over
10: Generating and transmitting a Pause control packet (in Full-duplex mode)
11: Transferring input packet for transmission

Bit 16 **TPESTS**: MAC MII Transmit Protocol Engine Status

When this bit is set, it indicates that the MAC MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **RFCFCSTS[1:0]**: MAC Receive Packet Controller FIFO Status

When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.

Bit 0 **RPESTS**: MAC MII Receive Protocol Engine Status

When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.

HW feature 0 register (ETH_MACHWF0R)

Address offset: 0x011C

Reset value: 0x0A0D 73F5

This register indicates the presence of first set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]		MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	r	r	r			r	r	r	r	r	r	r	r	r	r

Bits 31:28 **ACTPHYSEL[3:0]**: Active PHY selected
When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion:
0x0: GMII or MII
0x1: RGMII
0x2: SGMII
0x3: TBI
0x4: RMII
0x5: RTBI
0x6: SMII
Others: Reserved, must not be used

Bit 27 **SAVLANINS**: Source Address or VLAN Insertion Enable
This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected

Bits 26:25 **TSSTSSEL[1:0]**: Timestamp System Time Source
This bit indicates the source of the Timestamp system time:
01: Internal
10: External
11: Both
00: Reserved, must not be used
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bit 24 **MACADR64SEL**: MAC Addresses 64-127 Selected
This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected

Bit 23 **MACADR32SEL**: MAC Addresses 32-63 Selected
This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected

Bits 22:18 **ADDMACADRSEL[4:0]**: MAC Addresses 1-31 Selected
This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected

Bit 17 Reserved, must be kept at reset value.

Bit 16 **RXCOESEL**: Receive Checksum Offload Enabled
This bit is set to 1 when the Enable Receive TCP/IP Checksum Check option is selected

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXCOESEL**: Transmit Checksum Offload Enabled
This bit is set to 1 when the Enable Transmit TCP/IP Checksum Insertion option is selected

Bit 13 **EEESEL**: Energy Efficient Ethernet Enabled
This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected

Bit 12 **TSSEL**: IEEE 1588-2008 Timestamp Enabled
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ARPOFFSEL**: ARP Offload Enabled
This bit is set to 1 when the Enable IPv4 ARP Offload option is selected

Bit 8 **MMCSEL**: RMON Module Enable
This bit is set to 1 when the Enable MAC management counters (MMC) option is selected

Bit 7 **MGKSEL**: PMT Magic Packet Enable
This bit is set to 1 when the Enable Magic Packet Detection option is selected

Bit 6 **RWKSEL**: PMT Remote Wake-up Packet Enable
This bit is set to 1 when the Enable Remote Wake-up Packet Detection option is selected

Bit 5 **SMASEL**: SMA (MDIO) Interface
This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected

Bit 4 **VLHASH**: VLAN Hash Filter Selected
This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected

Bit 3 **PCSSEL**: PCS Registers (TBI, SGMII, or RTBI PHY interface)
This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected

Bit 2 **HDSEL**: Half-duplex Support
This bit is set to 1 when the Half-duplex mode is selected

Bit 1 **GMIISEL**: 1000 Mbps Support
This bit is set to 1 when 1000 Mbps is selected as operating mode.

Bit 0 **MIISEL**: 10 or 100 Mbps Support
This bit is set to 1 when 10/100 Mbps is selected as operating mode.

HW feature 1 register (ETH_MACHWF1R)

Address offset: 0x0120

Reset value: 0x1104 1904

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN
	r	r	r	r		r	r	r		r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					Res.	RXFIFOSIZE[4:0]				
r	r	r	r	r	r	r	r	r	r		r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
This field indicates the total number of L3 or L4 filters:
0000: No L3 or L4 Filter
0001: 1 L3 or L4 Filter
0010: 2 L3 or L4 Filters
..
1000: 8 L3 or L4

Bit 26 Reserved, must be kept at reset value.

Bits 25:24 **HASHTBLSZ[1:0]**: Hash Table Size
This field indicates the size of the Hash table:
00: No Hash table
01: 64
10: 128
11: 256

Bit 23 **POUOST**: One Step for PTP over UDP/IP Feature Enable
This bit is set to 1 when the Enable one step timestamp for PTP over UDP/IP feature is selected.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **RAVSEL**: Rx Side Only AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option on Rx Side Only is selected.

Bit 20 **AVSEL**: AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option is selected.

Bit 19 **DBGMEMA**: DMA Debug Registers Enable
This bit is set to 1 when the Debug Mode Enable option is selected

Bit 18 **TSOEN**: TCP Segmentation Offload Enable
This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Bit 17 **SPHEN**: Split Header Feature Enable
This bit is set to 1 when the Enable Split Header Structure option is selected

Bit 16 **DCBEN**: DCB Feature Enable
This bit is set to 1 when the Enable Data Center Bridging option is selected

Bits 15:14 **ADDR64[1:0]**: Address width
This field indicates the configured address width.
00: 32 bits
Others: Reserved, must not be used

Bit 13 **ADVTHWORD**: IEEE 1588 High Word Register Enable
This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected

Bit 12 **PTOEN**: PTP Offload Enable
This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11 **OSTEN**: One-Step Timestamping Enable
This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.

Bits 10:6 **TXFIFOSIZE[4:0]**: MTL Transmit FIFO Size
This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011 to 11111: Reserved, must not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **RXFIFOSIZE[4:0]**: MTL Receive FIFO Size
This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011: 256 Kbytes
01100 to 11111: Reserved, must not be used

HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124

Reset value: 0x4100 0000

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **AUXSNAPNUM[2:0]**: Number of Auxiliary Snapshot Inputs
This field indicates the number of auxiliary snapshot inputs:
000: No auxiliary input
001: 1 auxiliary input
010: 2 auxiliary inputs
011: 3 auxiliary inputs
100: 4 auxiliary inputs
101 to 111: Reserved, must not be used

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **PPSOUTNUM[2:0]**: Number of PPS Outputs
This field indicates the number of PPS outputs:
000: No PPS output
001: 1 PPS output
010: 2 PPS outputs
011: 3 PPS outputs
100: 4 PPS outputs
101 to 111: Reserved, must not be used

Bits 23:22 **TDCSZ[1:0]**: Tx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 21:18 **TXCHCNT[3:0]**: Number of DMA Transmit Channels
This field indicates the number of DMA Transmit channels:
0000: 1 DMA Tx Channel
0001: 2 DMA Tx Channels
..
0111: 8 DMA Tx

Bits 17:16 **RDCSZ[1:0]**: Rx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 15:12 **RXCHCNT[3:0]**: Number of DMA Receive Channels
This field indicates the number of DMA Receive channels:
0000: 1 DMA Rx Channel
0001: 2 DMA Rx Channels
..
0111: 8 DMA Rx

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **TXQCNT[3:0]**: Number of MTL Transmit Queues
This field indicates the number of MTL Transmit queues:
0000: 1 MTL Tx queue
0001: 2 MTL Tx queues
..
0111: 8 MTL Tx

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **RXQCNT[3:0]**: Number of MTL Receive Queues
This field indicates the number of MTL Receive queues:
0000: 1 MTL Rx queue
0001: 2 MTL Rx queues
..
0111: 8 MTL Rx

HW feature 3 register (ETH_MACHWF3R)

Address offset: 0x0128

Reset value: 0x0000 0020

This register indicates the presence of fourth set the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

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

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

Bit 5 **DVLAN**: Double VLAN processing enable
This bit is set to 1 when Double VLAN processing is enabled.

Bit 4 **CBTISEL**: Queue/Channel based VLAN tag insertion on Tx enable
This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx feature is selected.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **NRVF[2:0]**: Number of Extended VLAN Tag Filters Enabled
This field indicates the Number of Extended VLAN Tag Filters selected:
000: No Extended Rx VLAN Filters
001: 4 Extended Rx VLAN Filters
010: 8 Extended Rx VLAN Filters
011: 16 Extended Rx VLAN Filters
100: 24 Extended Rx VLAN Filters
101: 32 Extended Rx VLAN Filters
110 to 111: Reserved, must not be used

MDIO address register (ETH_MACMDIOAR)

Address offset: 0x0200

Reset value: 0x0000 0000

The MDIO Address register controls the management cycles to external PHY through a management interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]				
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	MB
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PSE**: Preamble Suppression Enable

When this bit is set, the SMA suppresses the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.

When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.

Bit 26 **BTB**: Back to Back transactions

When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame.

When this bit is reset, then the read/write command completion (MII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.

This bit must not be set when NTC=0.

Bits 25:21 **PA[4:0]**: Physical Layer Address

This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.
This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Bits 20:16 **RDA[4:0]**: Register/Device Address

These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **NTC[2:0]**: Number of Training Clocks

This field controls the number of trailing clock cycles generated on ETH_MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 011 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.

Bits 11:8 **CR[3:0]**: CSR Clock Range

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency (eth_hclk):
0000: MDC clock = eth_hclk / 42
0001: MDC clock = eth_hclk / 62
0010: MDC clock = eth_hclk / 16
0011: MDC clock = eth_hclk / 26
0100: MDC clock = eth_hclk / 102
0101: MDC clock = eth_hclk / 124
0110 to 0111: Reserved, must not be used
The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range.
When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits to 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks:
1000: eth_hclk / 4
1001: eth_hclk / 6
1010: eth_hclk / 8
1011: eth_hclk / 10
1100: eth_hclk / 12
1101: eth_hclk / 14
1110: eth_hclk / 16
1111: eth_hclk / 18

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SKAP**: Skip Address Packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Bits 3:2 **GOC[1:0]**: MII Operation Command

This bit indicates the operation command to the PHY.
00: Reserved, must not be used
01: Write
10: Post Read Increment Address for Clause 45 PHY
11: Read
When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E**: Clause 45 PHY Enable

When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.

Bit 0 **MB**: MII Busy

The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in *MDIO address register (ETH_MACMDIOAR)* and *MDIO data register (ETH_MACMDIODR)* as long as this bit is set.

For write transfers, the application must first write 16-bit data in the MD field (and also RA field when C45E is set) in *MDIO data register (ETH_MACMDIODR)* register before setting this bit. When C45E is set, it should also write into the RA field of *MDIO data register (ETH_MACMDIODR)* before initiating a read transfer. When a read transfer is completed (MII busy=0), the data read from the PHY register is valid in the MD field of the *MDIO data register (ETH_MACMDIODR)*.

Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit.

MDIO data register (ETH_MACMDIODR)

Address offset: 0x0204

Reset value: 0x0000 0000

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in *MDIO address register (ETH_MACMDIOAR)* This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

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

Bits 31:16 **RA[15:0]**: Register Address

This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

Bits 15:0 **MD[15:0]**: MII Data

This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

ARP address register (ETH_MACARPAR)

Address offset: 0x0210

Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

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

Bits 31:0 **ARPPA[31:0]**: ARP Protocol Address

This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

CSR software control register (ETH_MACCSRSWCR)

Address offset: 0x0230

Reset value: 0x0000 0000

This register contains software-programmable controls for changing the CSR access response and status bits clearing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SEEN**: Slave Error Response Enable

When this bit is set, the MAC responds with a Slave Error for accesses to reserved registers in CSR space.

When this bit is reset, the MAC responds with an Okay response to any register accessed from CSR space.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RCWE**: Register Clear on Write 1 Enable

When this bit is set, the access mode to some register fields changes to rc_w1 (clear on write) meaning that the application needs to set that respective bit to 1 to clear it.

When this bit is reset, the access mode to these register fields remains rc_r (clear on read).

MAC Address 0 high register (ETH_MACA0HR)

Address offset: 0x0300

Reset value: 0x8000 FFFF

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x11 22 33 44 55 66 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x66 55 44 33 22 11.

If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable
This bit is always set to 1.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address0[47:32]
This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)

Reset value: 0xFFFF FFFF

The MAC Address x Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)

This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x high register (ETH_MACAxHR)

Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)

Reset value: 0x0000 FFFF

The MAC Address x High register holds the upper 16 bits of the second 6-byte MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30 **SA**: Source Address

When this bit is set, the MAC Addressx[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.

0: DA

1: SA

Bits 29:24 **MBC[5:0]**: Mask Byte Control

These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:

Bit 29: ETH_MACAxHR[15:8]

Bit 28: ETH_MACAxHR[7:0]

Bit 27: ETH_MACAxLR[31:24]

Bit 26: ETH_MACAxLR[23:16]

Bit 25: ETH_MACAxLR[15:8]

Bit 24: ETH_MACAxLR[7:0]

You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:16 Reserved, must be kept at reset value.

Bits 15:0 **ADDRHI[15:0]**: MAC Address1 [47:32]

This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MMC control register (ETH_MMC_CONTROL)

Address offset: 0x0700

Reset value: 0x0000 0000

This register configures the MMC operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
							rw			rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets

The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of *Packet filtering control register (ETH_MACPFR)*.

When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset

When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF FFF0 (Half 16).

When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF F800 (Full 2Kbytes) and all packet-counters get preset to 0xFFFF FFF0 (Full 16).

For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.

Bit 4 **CNTPRST**: Counters Preset

When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.

This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze

When this bit is set, it freezes all MMC counters to their current value.

Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Bit 2 **RSTONRD**: Reset on Read

When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover

When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)

Address offset: 0x0704

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Receive statistics counters.

The MMC receive interrupt register maintains the interrupts that are generated when the following occur:

- Receive statistic counters reach half of their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, interrupts are set but the counter remains at all-ones. The MMC receive interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.
				rc_r	rc_r									rc_r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
									rc_r	rc_r					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIS**: MMC Receive LPI transition counter interrupt status

This bit is set when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIS**: MMC Receive LPI microsecond counter interrupt status

This bit is set when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIS**: MMC Receive Unicast Good Packet Counter Interrupt Status
This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIS**: MMC Receive Alignment Error Packet Counter Interrupt Status
This bit is set when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIS**: MMC Receive CRC Error Packet Counter Interrupt Status
This bit is set when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)

Address offset: 0x0708

Reset value: 0x0000 0000

This register maintains the interrupts generated from all Transmit statistics counters.

The MMC transmit interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones.

The MMC transmit interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.
				rc_r	rc_r					rc_r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rc_r	rc_r														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIS**: MMC Transmit LPI transition counter interrupt status
This bit is set when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIS**: MMC Transmit LPI microsecond counter interrupt status
This bit is set when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIS**: MMC Transmit Good Packet Counter Interrupt Status
This bit is set when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIS**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Status
This bit is set when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIS**: MMC Transmit Single Collision Good Packet Counter Interrupt Status
This bit is set when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

MMC Rx interrupt mask register (ETH_MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C

Reset value: 0x0000 0000

The MMC receive interrupt mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.
				rw	rw									rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
									rw	rw					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIM**: MMC Receive LPI transition counter interrupt Mask

Setting this bit masks the interrupt when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIM**: MMC Receive LPI microsecond counter interrupt Mask

Setting this bit masks the interrupt when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIM**: MMC Receive Unicast Good Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIM**: MMC Receive Alignment Error Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIM**: MMC Receive CRC Error Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt mask register (ETH_MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710

Reset value: 0x0000 0000

The MMC transmit interrupt mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32-bit wide.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.
				rw	rw					rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIM**: MMC Transmit LPI transition counter interrupt Mask

Setting this bit masks the interrupt when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIM**: MMC Transmit LPI microsecond counter interrupt Mask

Setting this bit masks the interrupt when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIM**: MMC Transmit Good Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIM**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIM**: MMC Transmit Single Collision Good Packet Counter Interrupt Mask

Setting this bit masks the interrupt when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXSNGLCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXSNGLCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXSNGLCOLG[31:0]**: Tx Single Collision Good Packets

This field indicates the number of successfully transmitted packets after a single collision in the Half-duplex mode.

Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMULTCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMULTCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMULTCOLG[31:0]**: Tx Multiple Collision Good Packets

This field indicates the number of successfully transmitted packets after multiple collisions in the Half-duplex mode.

Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)

Address offset: 0x0768

Reset value: 0x0000 0000

This register provides the number of good packets transmitted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPKTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPKTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPKTG[31:0]**: Tx Packet Count Good
This field indicates the number of good packets transmitted.

Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)

Address offset: 0x0794

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with CRC error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRCERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRCERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRCERR[31:0]**: Rx CRC Error Packets
This field indicates the number of packets received with CRC error.

Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)

Address offset: 0x0798

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXALGNERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXALGNERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXALGNERR[31:0]**: Rx Alignment Error Packets

This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)

Address offset: 0x07C4

Reset value: 0x0000 0000

This register provides the number of good unicast packets received by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXUCASTG[31:0]**: Rx Unicast Packets Good

This field indicates the number of good unicast packets received.

Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)

Address offset: 0x07EC

Reset value: 0x0000 0000

This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPIUSC[31:0]**: Tx LPI Microseconds Counter

This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)

Address offset: 0x07F0

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPITRC[31:0]**: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate mode (because of LPITXA bit set in the *LPI control and status register (ETH_MACLCSR)*), the counter is incremented.

Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)

Address offset: 0x07F4

Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPIUSC[31:0]**: Rx LPI Microseconds Counter

This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)

Address offset: 0x07F8

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPITRC[31:0]**: Rx LPI Transition counter
This field indicates the number of times Rx LPI Entry has occurred.

L3 and L4 control 0 register (ETH_MACL3L4C0R)

Address offset: 0x0900

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0
										rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM0**: Layer 4 Destination Port Inverse Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.
This bit is valid and applicable only when the L4DPM0 bit is set high.

Bit 20 **L4DPM0**: Layer 4 Destination Port Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM0**: Layer 4 Source Port Inverse Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.
This bit is valid and applicable only when the L4SPM0 bit is set high.

Bit 18 **L4SPM0**: Layer 4 Source Port Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN0**: Layer 4 Protocol Enable
When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.
The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of higher bits of IP Destination Address that are masked in the IPv4 packets:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. Number of bits masked is given by concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP source address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP source or destination address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.

Bit 2 **L3SAM0**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN0**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

Layer4 Address filter 0 register (ETH_MACL4A0R)

Address offset: 0x0904

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP0[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP0[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 Address 0 filter 0 register (ETH_MACL3A00R)

Address offset: 0x0910

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A00[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A00[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A00[31:0]**: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 Address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A10[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A10[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A10[31:0]**: Layer 3 Address 1 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 Address 2 filter 0 register (ETH_MACL3A20R)

Address offset: 0x0918

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A20[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A20[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A20[31:0]**: Layer 3 Address 2 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

Layer3 Address 3 filter 0 register (ETH_MACL3A30R)

Address offset: 0x091C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A30[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A30[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A30[31:0]**: Layer 3 Address 3 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

L3 and L4 control 1 register (ETH_MACL3L4C1R)

Address offset: 0x0930

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 1 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1
										rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM1**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM1 bit is set high.

Bit 20 **L4DPM1**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM1**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM1 bit is set high.

Bit 18 **L4SPM1**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN1**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM1 or L4DPM1 bit is set.

Bits 15:11 **L3HDBM1[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP Destination Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits:

0: No bits are masked

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

127: All bits except MSb are masked

This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set.

Bits 10:6 **L3HSBM1[4:0]**: Layer 3 IP SA Higher Bits Match

Condition: IPv4 packets

This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

This field contains Bits[4:0] of L3HSBM1. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set high.

Bit 5 **L3DAIM1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM1 bit is set high.

Bit 4 **L3DAM1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN1 bit is set, you should set either this bit or the L3SAM1 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM1**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM1 bit is set.

Bit 2 **L3SAM1**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN01 bit is set, you should set either this bit or the L3DAM1 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN1**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM1 or L3DAM1 bit is set.

Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP1[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP1[15:0]**: Layer 4 Source Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 1 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A01[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A01[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A01[31:0]**: Layer 3 Address 0 Field

When the L3PEN1 and L3SAM1bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 1 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A11[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A11[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 1 Register (ETH_MACL3A21R)

Address offset: 0x0948

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A21[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A21[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A21[31:0]**: Layer 3 Address 2 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

Layer3 address 3 filter 1 register (ETH_MACL3A31R)

Address offset: 0x94C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A31[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A31[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A31[31:0]**: Layer 3 Address 3 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

Timestamp control Register (ETH_MACTSCR)

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the receiver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]	
			rw				rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	Res.	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw			rw		rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **AV8021ASMEN**: AV 802.1AS Mode Enable

When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS operating mode.

When the PTP offload feature is enabled, for PTP offload, the transport-specific field in the PTP header is generated and checked based on the value of this bit.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TXTSSTSM**: Transmit Timestamp Status Mode

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering

When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.

When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.

For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.

For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTYPSEL[1:0]**: Select PTP packets for Taking Snapshots

These bits, along with Bits 15 and 14, define the set of PTP packet types for which snapshot needs to be taken. The encoding is given in *Table 655: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master

When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages

When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see *Table 655: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 13 **TSIPV4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPV6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets

When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format

When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control

When this bit is set, the Timestamp Low register rolls over after 0x3B9A C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of subsecond register is 0x7FFF FFFF. The subsecond increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets

When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **TSADDREG**: Update Addend Register

When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TSUPDT**: Update Timestamp

When this bit is set, the system time is updated (added or subtracted) with the value specified in *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.
This bit should be zero before updating it. This bit is reset when the update is complete in hardware.

Bit 2 **TSINIT**: Initialize Timestamp

When this bit is set, the system time is initialized (overwritten) with the value specified in the *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.

This bit should be zero before it is updated. This bit is reset when the initialization is complete.

Bit 1 **TSCFUPDT**: Fine or Coarse Timestamp Update

When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0 **TSENA**: Enable Timestamp

When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.
On the Receive side, the MAC processes the 1588 packets only if this bit is set.

Subsecond increment register (ETH_MACSSIR)

Address offset: 0x0B04

Reset value: 0x0000 0000

In Coarse Update mode (bit TSCFUPDT in *Timestamp control Register (ETH_MACTSCR)*), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SSINC[7:0]**: Subsecond Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the subsecond register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [TSCTRLSSR bit is set in *Timestamp control Register (ETH_MACTSCR)*]. When TSCTRLSSR is cleared, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:0 Reserved, must be kept at reset value.

System time seconds register (ETH_MACSTSR)

Address offset: 0x0B08

Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TSS[31:0]**: Timestamp Second

The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

System time nanoseconds register (ETH_MACSTNR)

Address offset: 0x0B0C

Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSSS[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field has the subsecond representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set in *Timestamp control Register (ETH_MACTSCR)*, each bit represents 1 ns. The maximum value is 0x3B9A C9FF after which it rolls-over to zero.

System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10

Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in *Timestamp control Register (ETH_MACTSCR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSS[31:0]**: Timestamp Seconds

The value in this field is the seconds part of the update.
When ADDSUB is reset, this field must be programmed with the seconds part of the update value.
When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value.
For example, to subtract 2.000000001 seconds from the system time, the TSS field in the ETH_MACSTSUR register must be 0xFFFF FFFE (that is, $2^{32} – 2$).

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0 in *System time nanoseconds update register (ETH_MACSTNUR)*. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, $2^{32} – 1$) and the *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] field = 0x7FFF FFFF)

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD SUB	TSSS[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADDSUB**: Add or Subtract Time

When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field is the subseconds part of the update.

– ADDSUB is 1: This field must be programmed with the complement of the subseconds part of the update value as described.

– ADDSUB is 0: This field must be programmed with the subseconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the *Timestamp control Register (ETH_MACTSCR)*.

– TSCTRLSSR field in the *Timestamp control Register (ETH_MACTSCR)*is 1:
- The programmed value must be 10^9 – <subsecond value>.
- Each bit represents 1 ns and the programmed value should not exceed 0x3B9A C9FF.

– TSCTRLSSR field in the *Timestamp control Register (ETH_MACTSCR)* is 0:
- The programmed value must be 2^{31} - <subsecond_value>.
- Each bit represents an accuracy of 0.46 ns.

For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the ETH_MACSTNUR register must be 0x7FFF FFFF (that is, 2^{31} – 1), when TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset and 0x3B9A C9FF (that is, 10^9 –1), when TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set.

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] = 0) and the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, 2^{32} – 1).

Timestamp addend register (ETH_MACTSAR)

Address offset: 0x0B18

Reset value: 0x0000 0000

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the *Timestamp control Register (ETH_MACTSCR)*). The content of this register is added to a 32-bit accumulator in every clock cycle of clk_ptp_ref_i and the system time is updated whenever the accumulator overflows.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSAR[31:0]**: Timestamp Addend Register

This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Timestamp status register (ETH_MACTSSR)

Address offset: 0x0B20

Reset value: 0x0000 0000

All bits except Bits[27:25] gets cleared when the application reads this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]			
		r	r	r	r	r	rc_r					rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
rc_r												rc_r	rc_r	rc_r	rc_r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **ATSNS[4:0]**: Number of Auxiliary Timestamp Snapshots

This field indicates the number of Snapshots available in the FIFO. A value equal to the depth of FIFO (4) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.

Bit 24 **ATSSTM**: Auxiliary Timestamp Snapshot Trigger Missed

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATSSTN[3:0]**: Auxiliary Timestamp Snapshot Trigger Identifier

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

Bit 16: Auxiliary trigger 0

Bit 17: Auxiliary trigger 1

Bit 18: Auxiliary trigger 2

Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

Bit 15 **TXTSSIS**: Tx Timestamp Status Interrupt Status

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the *Tx timestamp status seconds register (ETH_MACTXTSSSR)* is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 14:4 Reserved, must be kept at reset value.

Bit 3 **TSTRGTERR0**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR elapses (see *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*). This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **AUXTSTRIG**: Auxiliary Timestamp Trigger Snapshot

This bit is set high when the auxiliary snapshot is written to the FIFO.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **TSTARGT0**: Timestamp Target Time Reached

When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR registers (see *PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*).
This bit is cleared when the application reads this bit (or writes of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Bit 0 **TSSOVF**: Timestamp Seconds Overflow

When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 0xFFFF FFFF.
This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)

Address offset: 0x0B30

Reset value: 0x0000 0000

This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSMIS	TXTSSLO[30:16]														
r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSLO[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bit 31 **TXTSSMIS**: Transmit Timestamp Status Missed

When this bit is set, it indicates one of the following:

– The timestamp of the current packet is ignored if TXTSSTSM bit of the *Timestamp control Register (ETH_MACTSCR)* is reset

– The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the *Timestamp control Register (ETH_MACTSCR)* is set.

Bits 30:0 **TXTSSLO[30:0]**: Transmit Timestamp Status Low

This field contains the 31 bits of the Nanoseconds field of the Transmit packet's captured timestamp.

Tx timestamp status seconds register (ETH_MACTXTSSSR)

Address offset: 0x0B34

Reset value: 0x0000 0000

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXTSSHI[31:0]**: Transmit Timestamp Status High

This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

Auxiliary control register (ETH_MACACR)

Address offset: 0x0B40

Reset value: 0x0000 0000

The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATSEN3**: Auxiliary Snapshot 3 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg3 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 6 **ATSEN2**: Auxiliary Snapshot 2 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg2 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 5 **ATSEN1**: Auxiliary Snapshot 1 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg1 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg0 input is enabled. When this bit is reset, the events on this input are ignored.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear

When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

Auxiliary timestamp nanoseconds register (ETH_MACATSNR)

Address offset: 0x0B48

Reset value: 0x0000 0000

The *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*, along with *Auxiliary timestamp seconds register (ETH_MACATSSR)*, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in *Timestamp status register (ETH_MACTSSR)* indicate the fill-level of the FIFO. The top of the FIFO is removed only when *Auxiliary timestamp seconds register (ETH_MACATSSR)* is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXTSLO[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSLO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp

Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

Auxiliary timestamp seconds register (ETH_MACATSSR)

Address offset: 0x0B4C

Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUXTSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUXTSHI[31:0]**: Auxiliary Timestamp

Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

Timestamp ingress asymmetric correction register (ETH_MACTSIACR)

Address offset: 0x0B50

Reset value: 0x0000 0000

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTIAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTIAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction

This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000.

The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Timestamp egress asymmetric correction register (ETH_MACTSEACR)

Address offset: 0x0B54

Reset value: 0x0000 0000

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTEAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTEAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction

This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}.

For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD 8000, which is the 2's complement of 0x0002 8000(2.5×2^{16}). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003 4CCC (3.3×2^{16}).

Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)

Address offset: 0x0B58

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSIC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSIC[31:0]**: Timestamp Ingress Correction

This field contains the ingress path correction value as defined by the Ingress Correction expression.

Timestamp egress correction nanosecond register (ETH_MACTSECNR)

Address offset: 0x0B5C

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSEC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSEC[31:0]**: Timestamp Egress Correction

This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Reserved, must not be used

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, PPSCTRL[3:0] function as PPSCMD[3:0]. When this bit is reset, PPSCTRL[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCTRL[3:0]**: PPS Output Frequency Control

This field controls the frequency of the PPS output (eth_ptp_pps_out) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:

0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.

0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.

0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.

0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

..

1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

Note: In the binary rollover mode, the PPS output (eth_ptp_pps_out) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- *When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.*
- *When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of*
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high).
- *When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of*
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.

PPS control register [alternate] (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS target time seconds register (ETH_MACPPSTTSR)* and *PPS target time nanoseconds register (ETH_MACPPSTTNR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Reserved, must not be used

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD[3:0]. When this bit is reset, Bits[3:0] function as PPSCTRL(Fixed PPS mode).

Bits 3:0 **PPSCMD[3:0]:** Flexible PPS Output (eth_ptp_pps_out) Control

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are ‘all zero’. The following list describes the values of PPSCMD0:

0000: No Command

0001: START Single Pulse.

This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.

0010: START Pulse Train.

This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.

0011: Cancel START.

This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.

0100: STOP Pulse Train at time.

This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010) after the time programmed in the Target Time registers elapses.

0101: STOP Pulse Train immediately.

This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010).

0110: Cancel STOP Pulse train.

This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.

0111 to 1111: Reserved, must not be used

PPS target time seconds register (ETH_MACPPSTTSR)

Address offset: 0x0B80

Reset value: 0x0000 0000

The PPS output Target Time Seconds register, along with *PPS target time nanoseconds register (ETH_MACPPSTTNR)*, is used to schedule an interrupt event (Bit TSSOVF of *Timestamp status register (ETH_MACTSSR)*) when the system time exceeds the value programmed in these registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSTRH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTRH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSTRH0[31:0]**: PPS Target Time Seconds Register

This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the *PPS control register (ETH_MACPPSCR)*.

PPS target time nanoseconds register (ETH_MACPPSTTNR)

Address offset: 0x0B84

Reset value: 0x0000 0000

The PPS Target Time Nanoseconds register is present only when more than one Flexible PPS output is selected.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRGTBUSY0	TTSL0[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TRGTBUSY0**: PPS Target Time Register Busy

The MAC sets this bit when the PPSCMD0 field in the *PPS control register (ETH_MACPPSCR)* is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers with the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

Bits 30:0 **TTSL0[30:0]**: Target Time Low for PPS Register

This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in *PPS control register (ETH_MACPPSCR)*.

When the TSCTRLSSR bit is set in the *Timestamp control Register (ETH_MACTSCR)*, this value should not exceed 0x3B9A C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of subsecond increment value.

PPS interval register (ETH_MACPPSIR)

Address offset: 0x0B88

Reset value: 0x0000 0000

The PPS Interval register contains the number of units of subsecond increment value between the rising edges of PPS output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSINT0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSINT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSINT0[31:0]**: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of subsecond increment value), you should program value 4 (5-1) in this register.

PPS width register (ETH_MACPPSWR)

Address offset: 0x0B8C

Reset value: 0x0000 0000

The PPS Width register contains the number of units of subsecond increment value between the rising and corresponding falling edges of PPS output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSWIDTH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSWIDTH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSWIDTH0[31:0]**: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of subsecond increment value), you should program value 3 (4-1) in this register.

Note: The value programmed in this register must be lesser than the value programmed in PPS interval register (ETH_MACPPSIR).

PTP Offload control register (ETH_MACPOCR)

Address offset: 0x0BC0

Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DN[7:0]								Res.	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DN[7:0]**: Domain Number
This field indicates the domain Number in which the PTP node is operating.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DRRDIS**: Disable PTO Delay Request/Response response generation
When this bit is set, the Delay Request and Delay response are not generated for received SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 **APDREQTRIG**: Automatic PTP Pdelay_Req message Trigger
When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the APDREQEN bit for this operation.

Bit 4 **ASYNCTRIG**: Automatic PTP SYNC message Trigger
When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the ASYNCEN bit for this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **APDREQEN**: Automatic PTP Pdelay_Req message Enable
When this bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer Transparent mode.

Bit 1 **ASYNCEN**: Automatic PTP SYNC message Enable
When this bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Clock Master mode.

Bit 0 **PTOEN**: PTP Offload Enable
When this bit is set, the PTP Offload feature is enabled.

PTP Source Port Identity 0 Register (ETH_MACSPI0R)

Address offset: 0x0BC4

Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI0[31:0]**: Source Port Identity 0
This field indicates bits [31:0] of sourcePortIdentity of PTP node.

PTP Source port identity 1 register (ETH_MACSPI1R)

Address offset: 0x0BC8

Reset value: 0x0000 0000

This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI1[31:0]**: Source Port Identity 1
This field indicates bits [63:32] of sourcePortIdentity of PTP node.

PTP Source port identity 2 register (ETH_MACSPI2R)

Address offset: 0x0BCC

Reset value: 0x0000 0000

This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SPI2[15:0]**: Source Port Identity 2
This field indicates bits [79:64] of sourcePortIdentity of PTP node.

Log message interval register (ETH_MACLMIR)

Address offset: 0x0BD0

Reset value: 0x0000 0000

This register contains the periodic intervals for automatic PTP packet generation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LMPDRI[7:0]**: Log Min Pdelay_Req Interval
This field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11 Reserved, must be kept at reset value.

Bits 10:8 **DRSYNCR[2:0]**:
Delay_Req to SYNC Ratio
In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.
0: DelayReq generated for every received SYNC
1: DelayReq generated every alternate reception of SYNC
2: for every 4 SYNC messages
3: for every 8 SYNC messages
4: for every 16 SYNC messages
5: for every 32 SYNC messages
Others: Reserved, must not be used
The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0 **LSI[7:0]**:
Log Sync Interval
This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Ethernet MAC register map and reset values

Table 702. Ethernet MAC register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	ETH_MACCR	ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	Res.	JD	JE	Res.	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0004	ETH_MACECR	Res.	Res.	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC	Res.	Res.	GPSL[13:0]														
	Reset value			0	0	0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	ETH_MACPFR	RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE	Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
	Reset value	0										0	0				0						0	0	0	0	0	0	0	0	0	0	0
0x000C	ETH_MACWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
	Reset value																								0					0	0	0	0
0x0010	ETH_MACHT0R	HT31T0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	ETH_MACHT1R	HT63T32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018 - 0x004C	Reserved																																
0x0050	ETH_MACVTR	EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV	VL[15:0]																
	Reset value	0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	Reserved																																
0x0058	ETH_MACVHTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLHT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	Reserved																																
0x0060	ETH_MACVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0064	ETH_MACiVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068 - 0x006C	Reserved																																
0x0070	ETH_ MACQTXFCR	PT[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0			0	0
0x0074 - 0x008C	Reserved																																
0x0090	ETH_MACRXFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
	Reset value																															0	0
0x0094- 0x00AC	Reserved																																
0x00B0	ETH_MACISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	Res.
	Reset value																		0	0	0		0	0	0			0	0	0			
0x00B4	ETH_MACIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	Res.
	Reset value																		0	0	0							0	0	0			
0x00B8	ETH_ MACRXTXSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
	Reset value																								0			0	0	0	0	0	0
0x00BC	Reserved																																
0x00C0	ETH_MACPCSR	RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
	Reset value	0			0	0	0	0	0														0	0			0	0			0	0	0
0x00C4	ETH_ MACRWKPFR	MACRWKPFR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8 - 0x00CC	Reserved																																

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00D0	ETH_MACLCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	Res.	PLS	LPIEN	Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
	Reset value											0	0	0		0	0							0	0					0	0	0	0
0x00D4	ETH_MACLTCR	Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]										TWT[15:0]															
	Reset value							1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	ETH_MACLETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	ETH_ MAC1USTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
	Reset value																					0	0	0	0	0	1	1	0	0	0	1	1
0x00E0 - 0x010C	Reserved																																
0x0110	ETH_MACVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERVER[7:0]								SNPSVER[7:0]							
	Reset value	0x0000 3242																															
0x0114	ETH_MACDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
	Reset value														0	0	0														0	0	0
0x0118	Reserved																																
0x011C	ETH_MACHWF0R	ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]			MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					RXCOESEL	Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	Reset value	0x0A0D 73F5																															
0x0120	ETH_MACHWF1R	Res.	L3L4FNUM[3:0]			Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN	ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]			Res.	RXFIFOSIZE[4:0]								
	Reset value	0x1104 1904																															

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0124	ETH_MACHWF2R	Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]		RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
	Reset value	0x4100 0000																															
0x0128	ETH_MACHWF3R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
	Reset value	0x0000 0020																															
0x012C - 0x01FC	Reserved																																
0x0200	ETH_ MACMDIOAR	Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]					Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	MB
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0
0x0204	ETH_ MACMDIODR	RA[15:0]																MD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0208 - 0x020C	Reserved																																
0x0210	ETH_MACARPAR	ARPPA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0214 - 0x022C	Reserved																																
0x0230	ETH_ MACCSRSWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
	Reset value																								0								0
0x0234- 0x23C	Reserved																																
0x0240 - 0x02FC	Reserved																																
0x0300	ETH_MACA0HR	AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	1																1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0304	ETH_MACA0LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0308	ETH_MACA1HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30C	ETH_MACA1LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0310	ETH_MACA2HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0314	ETH_MACA2LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0318	ETH_MACA3HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x031C	ETH_MACA3LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0320 - 0x06FC	Reserved																																
0x0700	ETH_MMC_ CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
	Reset value																								0			0	0	0	0	0	0
0x0704	ETH_MMC_RX_ INTERRUPT	Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0708	ETH_MMC_TX_ INTERRUPT	Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x070C	ETH_MMC_RX_INTERRUPT_MASK	Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0710	ETH_MMC_TX_INTERRUPT_MASK	Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x0714 - 0x0748	Reserved																																
0x074C	ETH_TX_SINGLE_COLLISION_GOOD_PACKETS	TXSNGLCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0750	ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS	TXMULTCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0754 - 0x0764	Reserved																																
0x0768	ETH_TX_PACKET_COUNT_GOOD	TXPKTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x076C - 0x0790	Reserved																																
0x0794	ETH_RX_CRC_ERROR_PACKETS	RXCRCERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0798	ETH_RX_ALIGNMENT_ERROR_PACKETS	RXALGNERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x079C - 0x07C0	Reserved																																
0x07C4	ETH_RX_UNICAST_PACKETS_GOOD	RXUCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C8 - 0x07E8	Reserved																																

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x07EC	ETH_TX_LPI_USEC_CNTR	TXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F0	ETH_TX_LPI_TRAN_CNTR	TXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F4	ETH_RX_LPI_USEC_CNTR	RXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F8	ETH_RX_LPI_TRAN_CNTR	RXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07FC-0x8FC	Reserved																																
0x0900	ETH_MACL3L4C0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0	L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0904	ETH_MACL4A0R	L4DP0[15:0]																L4SP0[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0908 - 0x090C	Reserved																																
0x0910	ETH_MACL3A00R	L3A00[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0914	ETH_MACL3A10R	L3A10[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0918	ETH_MACL3A20R	L3A20[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x091C	ETH_MACL3A30R	L3A30[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0920 - 0x092C	Reserved																																
0x0930	ETH_MACL3L4C1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1	L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
	Reset value											0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0934	ETH_MACL4A1R	L4DP1[15:0]																L4SP1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0938 - 0x093C	Reserved																																
0x0940	ETH_MACL3A01R	L3A01[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0944	ETH_MACL3A11R	L3A11[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0948	ETH_MACL3A21R	L3A21[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094C	ETH_MACL3A31R	L3A31[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0950 - 0x0AFC	Reserved																																
0x0B00	ETH_MACTSCR	Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]		TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	Res.	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
	Reset value				0				0						0	0	0	0	0	1	0	0	0	0	0			0		0	0	0	0
0x0B04	ETH_MACSSIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0																
0x0B08	ETH_MACSTSR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0C	ETH_MACSTNR	Res.	TSSS[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B10	ETH_MACSTSUR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B14	ETH_MACSTNUR	ADDSUB	TSSS[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B18	ETH_MACTSAR	TSAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B1C	Reserved																																

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B20	ETH_MACTSSR	Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]				TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
	Reset value			0	0	0	0	0	0					0	0	0	0	0												0	0	0	0
0x0B24-0xB28	Reserved																																
0x0B2C	Reserved																																
0x0B30	ETH_ MACTXTSSNR	TXTSSMIS	TXTSSLO[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B34	ETH_ MACTXTSSSR	TXTSSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B38 - 0x0B3C	Reserved																																
0x0B40	ETH_MACACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
	Reset value																									0	0	0	0				0
0x0B44	Reserved																																
0x0B48	ETH_MACATSNR	Res.	AUXTSLO[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4C	ETH_MACATSSR	AUXTSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B50	ETH_MACTSIACR	OSTIAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B54	ETH_MACTSEACR	OSTEAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B58	ETH_ MACTSICNR	TSIC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B5C	ETH_ MACTSECNR	TSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B60 - 0x0B6C	Reserved																																

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B70	ETH_MACPPSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
	Reset value																										0	0	0	0	0	0	0
0x0B70	ETH_MACPPSCR (alternate)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
	Reset value																										0	0	0	0	0	0	0
0x0B74 - 0x0B7C	Reserved																																
0x0B80	ETH_ MACPPSTTSR	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B84	ETH_ MACPPSTTNR	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B88	ETH_MACPPSIR	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8C	ETH_MACPPSWR	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B90 - 0x0BBC	Reserved																																
0x0BC0	ETH_MACPOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DN[7:0]								Res.	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
	Reset value																	0	0	0	0	0	0	0	0		0	0	0		0	0	0
0x0BC4	ETH_MACSPI0R	SPI0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC8	ETH_MACSPI1R	SPI1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 702. Ethernet MAC register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0BCC	ETH_MACSPI2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BD0	ETH_MACLMIR	LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
	Reset value	0	0	0	0	0	0	0	0														0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

58 HDMI-CEC controller (CEC)

58.1 HDMI-CEC introduction

Consumer electronics control (CEC) is part of HDMI (high-definition multimedia interface) standard. It contains a protocol that provides high-level control functions between various audiovisual products. CEC operates at low speeds, with minimum processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

58.2 HDMI-CEC controller main features

- Complies with HDMI-CEC v1.4 specification
- Independent 32 kHz CEC kernel (refer to *section RCC kernel clock distribution*)
- Works in Stop mode for ultra-low-power applications
- Configurable signal-free time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable peripheral address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration lost detection (ARBLST)
  - with automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)

58.3 HDMI-CEC functional description

58.3.1 HDMI-CEC pin and internal signals

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up must be added externally to the microcontroller.

To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This can be done by using a MOS transistor, as shown on *Figure 840*.

Table 704 lists the internal signals that are exchanged between the HDMI-CEC and other functional blocks (such as RCC and EXTI).

Table 703. HDMI pin

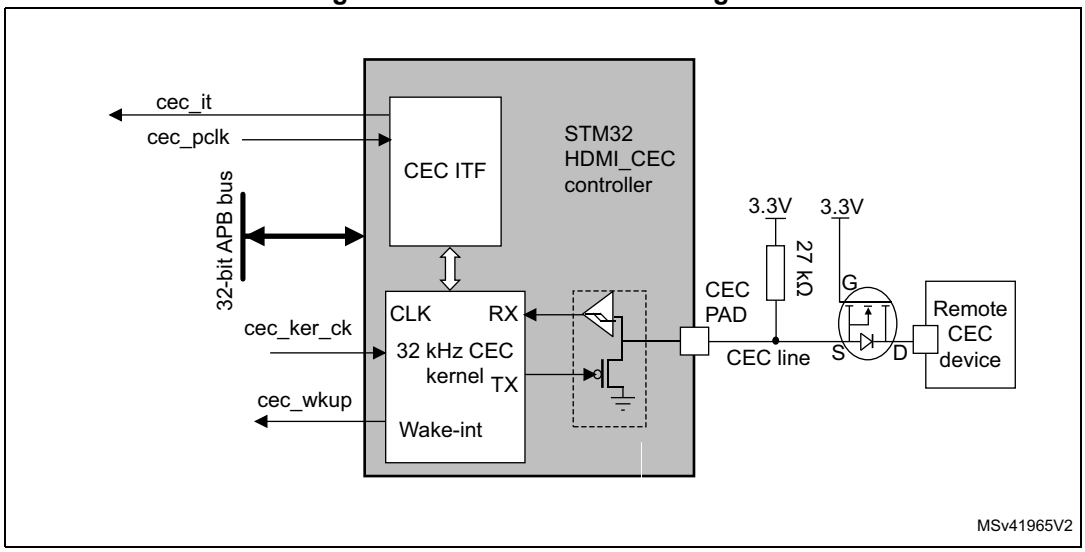
Name	Signal type	Remarks
CEC	Bidirectional	Two states: – 1 = high impedance – 0 = low impedance A 27 kΩ resistor must be added externally.

Table 704. HDMI-CEC internal input/output signals

Signal name	Signal type	Description
cec_wkup	Digital output	HDMI-CEC wake-up signal
cec_it	Digital output	HDMI-CEC interrupt signal
cec_pclk	Digital input	APB clock
cec_ker_ck	Digital input	HDMI-CEC kernel clock

58.3.2 HDMI-CEC block diagram

Figure 840. HDMI-CEC block diagram



58.3.3 Message description

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgment bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.

All these blocks are made of a 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In case a message contains additional blocks after an EOM is indicated, those additional blocks must be ignored. The EOM bit may be set in the header block to ‘ping’ other devices, to make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header, or by the follower that needs to reject a broadcast message.

The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

Figure 841. Message structure

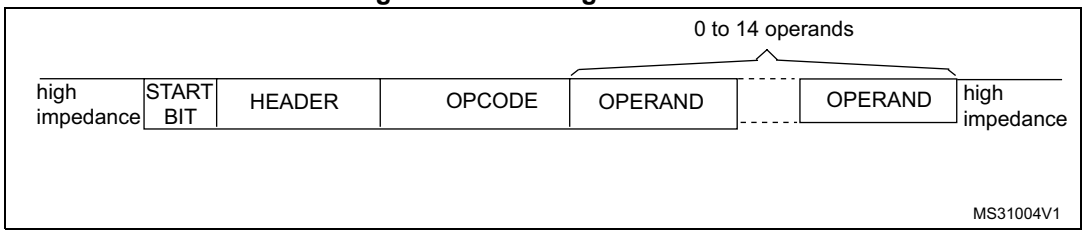
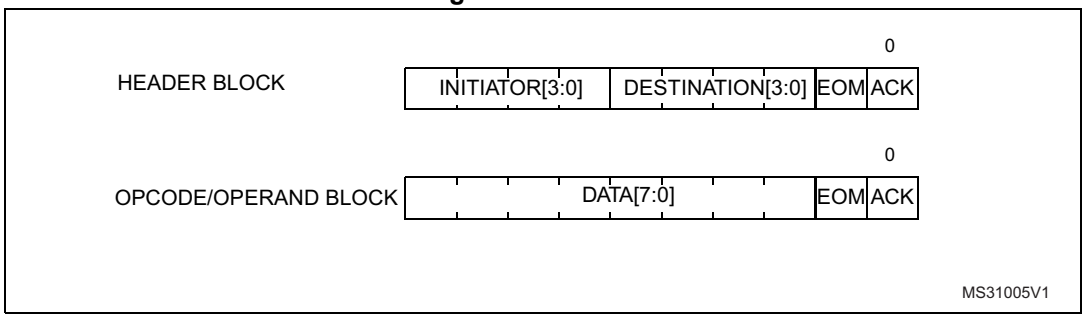


Figure 842. Blocks

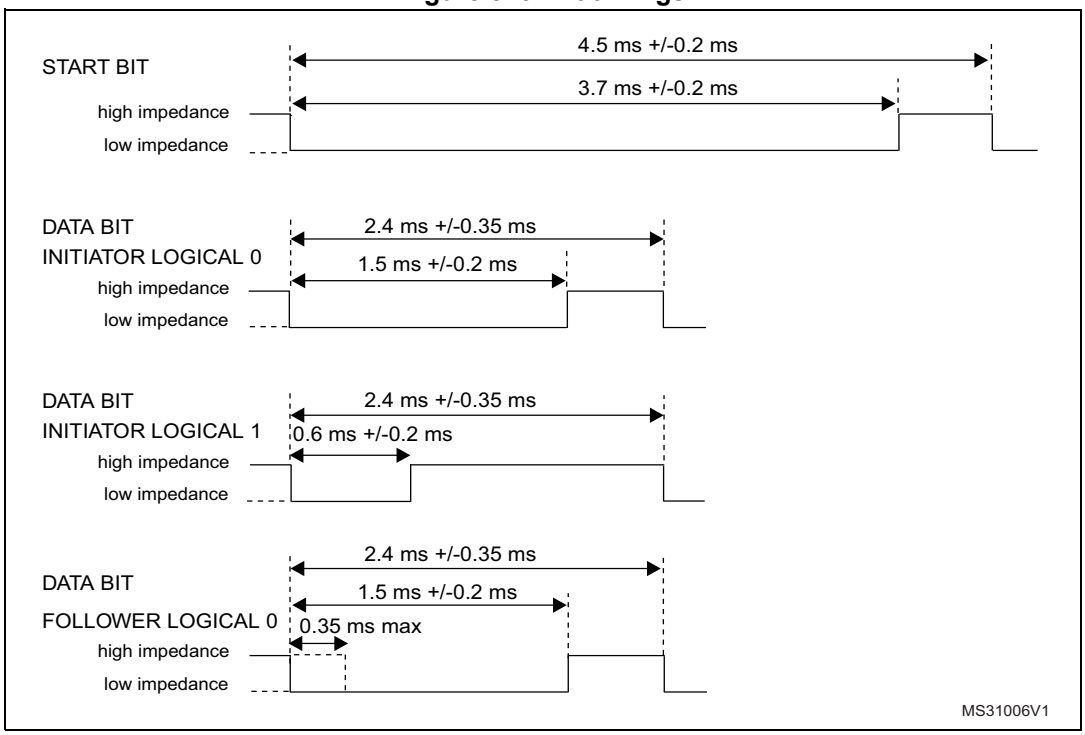


58.3.4 Bit timing

The format of the start bit is unique and identifies the start of a message. It must be validated by its low duration and its total duration.

All remaining data bits in the message, after the start bit, have consistent timing. The high-to-low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.

Figure 843. Bit timings

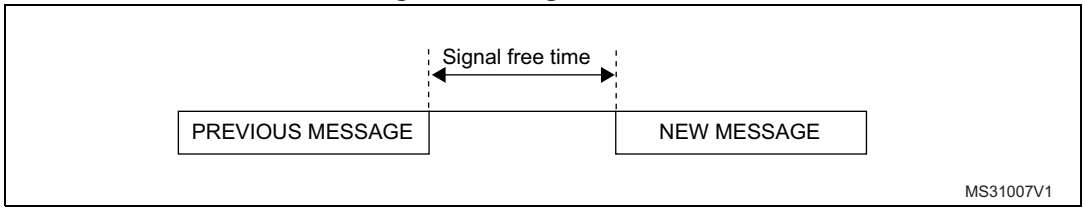


58.4 Arbitration

All devices transmitting - or retransmitting - a message onto the CEC line must ensure that it has been inactive for a number of bit periods. This signal-free time is defined as the time

starting from the final bit of the previous frame and depends on the initiating device and the current status as shown in the figure below.

Figure 844. Signal free time



Since only one initiator is allowed at any one time, an arbitration mechanism is provided to avoid conflict when more than one initiator begins transmitting at the same time.

CEC line arbitration starts with the leading edge of the start bit and continues until the end of the initiator address bits within the header block. During this period, the initiator must monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0. Assuming then it has lost arbitration, it stops transmitting and becomes a follower.

Figure 845. Arbitration phase

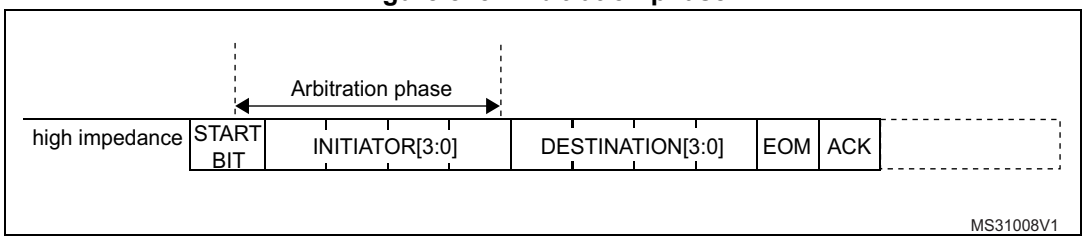
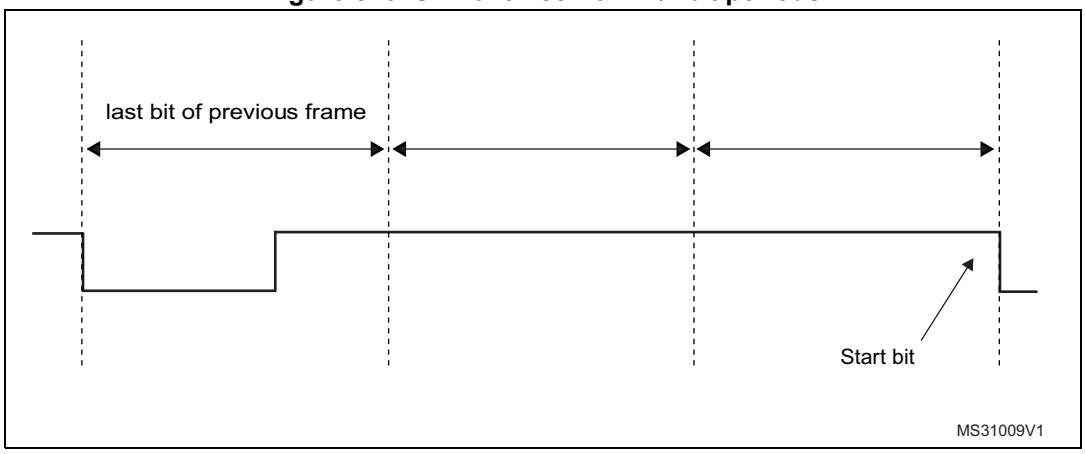


Figure 846 shows an example for a SFT of three nominal bit periods.

Figure 846. SFT of three nominal bit periods



A configurable time window is counted before starting the transmission.

In the SFT = 0 configuration, HDMI-CEC performs automatic SFT calculation ensuring compliance with the HDMI-CEC standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission

This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFT bits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

58.4.1 SFT option bit

In case of SFTOPT = 0 configuration, SFT starts being counted when the start-of-transmission command is set by software (TXSOM = 1).

In case of SFTOPT = 1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT timer is still running instead, the system waits until the timer elapses before transmission can start.

In case of SFTOPT = 1 a bus-event condition starting the SFT timer is detected in the following cases:

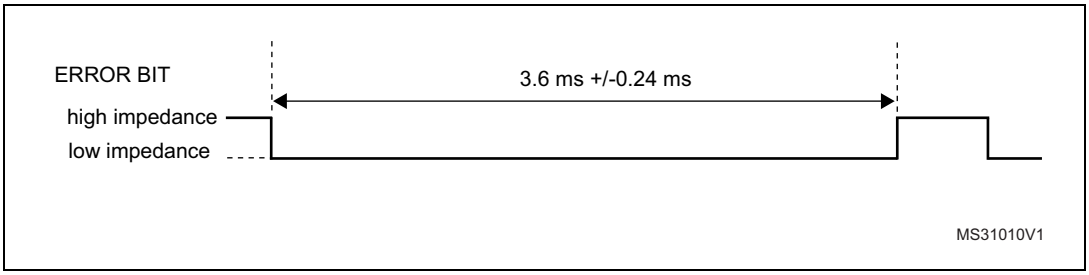
- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR = 1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACKE bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any uncodified low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

58.5 Error handling

58.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period (3.6 ms nominally).

Figure 847. Error bit timing



58.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

58.5.3 Bit rising error (BRE)

BRE (bit rising error) is set when a bit rising edge is detected outside the windows where it is expected (see *Figure 848*). BRE flag also generates a CEC interrupt if the BREIE = 1.

In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP = 1 an error bit is generated even if BREGEN = 0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN = 0, BRDNOGEN = 1.

58.5.4 Short bit period error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see *Figure 848*). SBPE flag also generates a CEC interrupt if the SBPEIE = 1.

An error bit is always generated on the line in case of a SBPE error detection. An error bit is not generated upon SBPE detection only when Listen mode is set (LSTN = 1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN = 1

58.5.5 Long bit period error (LBPE)

LBPE is set when a bit falling edge is not detected in a valid window (see *Figure 848*). LBPE flag also generates a CEC interrupt if the LBPEIE = 1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN = 0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN = 0, BRDNOGEN = 1.

Note: The BREGEN = 1, BRESTP = 0 configuration must be avoided.

Figure 848. Error handling

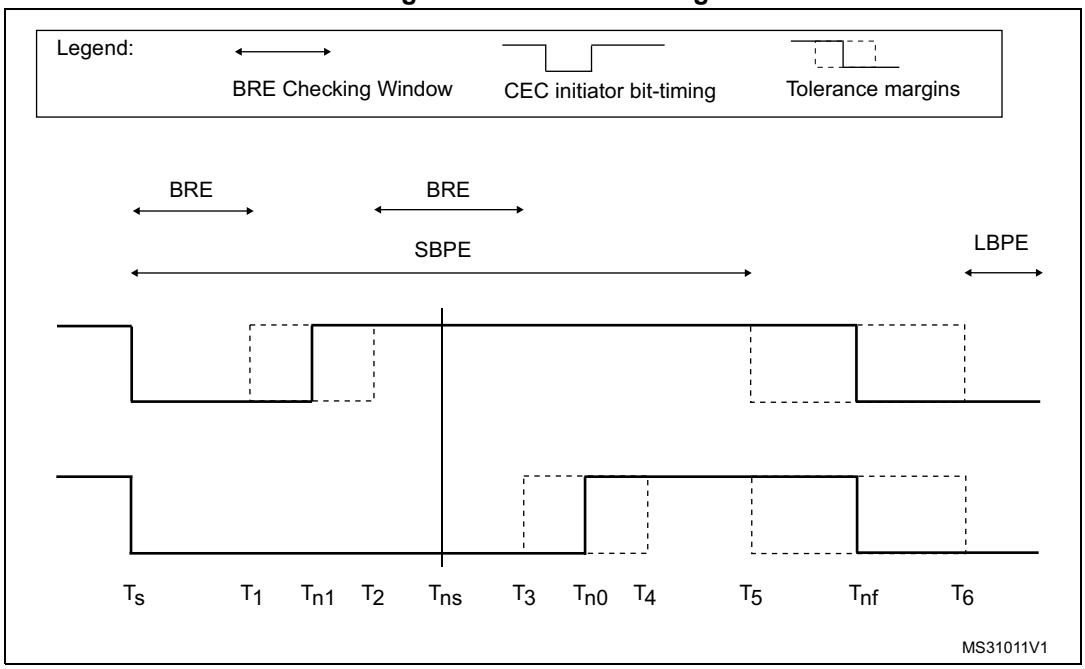


Table 705. Error handling timing parameters

Time	RXTOL	ms	Description
T_s	x	0	Bit start event.
T_1	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
T_{n1}	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
T_2	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
T_{ns}	x	1.05	Nominal sampling time.
T_3	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	
T_{n0}	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
T_4	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T_5	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T_{nf}	x	2.4	The nominal data bit period.
T_6	0	2.75	The latest time for the start of a following bit.
	1	2.95	

58.5.6 Transmission error detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE = 1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to five times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

Figure 849. TXERR detection

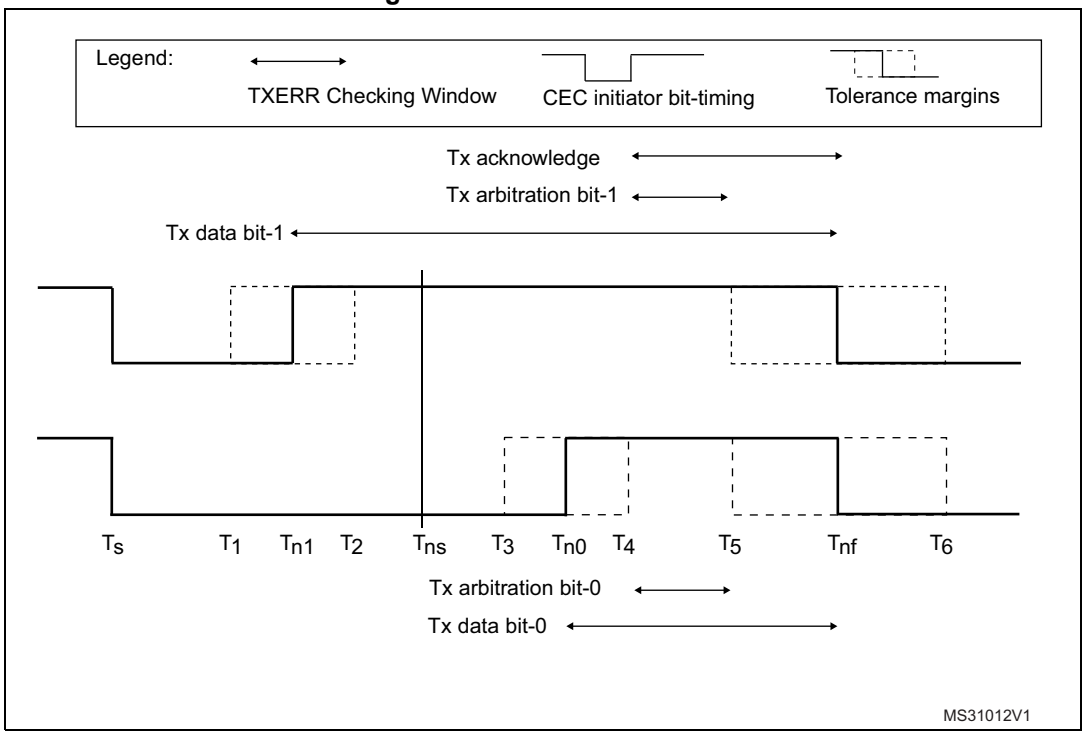


Table 706. TXERR timing parameters

Time	RXTOL	ms	Description
T_s	x	0	Bit start event.
T_1	1	0.3	The earliest time for a low - high transition when indicating a logical 1.
	0	0.4	
T_{n1}	x	0.6	The nominal time for a low - high transition when indicating a logical 1.
T_2	0	0.8	The latest time for a low - high transition when indicating a logical 1.
	1	0.9	
T_{ns}	x	1.05	Nominal sampling time.
T_3	1	1.2	The earliest time a device is permitted return to a high impedance state (logical 0).
	0	1.3	

Table 706. TXERR timing parameters (continued)

Time	RXTOL	ms	Description
T_{n0}	x	1.5	The nominal time a device is permitted return to a high impedance state (logical 0).
T_4	0	1.7	The latest time a device is permitted return to a high impedance state (logical 0).
	1	1.8	
T_5	1	1.85	The earliest time for the start of a following bit.
	0	2.05	
T_{nf}	x	2.4	The nominal data bit period.
T_6	0	2.75	The latest time for the start of a following bit.
	1	2.95	

58.6 HDMI-CEC interrupts

An interrupt can be produced:

- during reception if a receive block transfer is finished or if a receive error occurs.
- during transmission if a transmit block transfer is finished or if a transmit error occurs.

Table 707. HDMI-CEC interrupts

Interrupt event	Event flag	Enable control bit
Rx-byte received	RXBR	RXBRIE
End of reception	RXEND	RXENDIE
Rx-overrun	RXOVR	RXOVRIE
Rxbit rising error	BRE	BREIE
Rx-short bit period error	SBPE	SBPEIE
Rx-long bit period error	LBPE	LBPEIE
Rx-missing acknowledge error	RXACKE	RXACKEIE
Arbitration lost	ARBLST	ARBLSTIE
Tx-byte request	TXBR	TXBRIE
End of transmission	TXEND	TXENDIE
Tx-buffer underrun	TXUDR	TXUDRIE
Tx-error	TXERR	TXERRIE
Tx-missing acknowledge error	TXACKE	TXACKEIE

58.7 HDMI-CEC registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions. The registers have to be accessed by words (32 bits).

58.7.1 CEC control register (CEC_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEOM	TXSOM	CECEN
													rs	rs	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TXEOM***:* Tx end of message

The TXEOM bit is set by software to command transmission of the last byte of a CEC message.
TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.
0: TXDR data byte is transmitted with EOM = 0.
1: TXDR data byte is transmitted with EOM = 1.

Note: TXEOM must be set when CECEN = 1.

TXEOM must be set before writing transmission data to TXDR.

If TXEOM is set when TXSOM = 0, transmitted message consists of 1 byte (HEADER) only (PING message).

Bit 1 **TXSOM**: Tx start of message

TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM.
Start-bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission starts after the end of reception.
TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND = 1), in case of transmission underrun (TXUDR = 1), negative acknowledge (TXACKE = 1), and transmission error (TXERR = 1). It is also cleared by CECEN = 0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST = 1).
TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.
0: No CEC transmission is on-going.
1: CEC transmission command

Note: TXSOM must be set when CECEN = 1.

TXSOM must be set when transmission data is available into TXDR.

HEADER first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC_CFGR.OAR that is used only for reception.

Bit 0 **CECEN:** CEC enable

The CECEN bit is set and cleared by software. CECEN = 1 starts message reception and enables the TXSOM control. CECEN = 0 disables the CEC peripheral, clears all bits of CEC_CR register and aborts any on-going reception or transmission.

0: CEC peripheral is off.

1: CEC peripheral is on.

58.7.2 CEC configuration register (CEC_CFGR)

This register is used to configure the HDMI-CEC controller.

Address offset: 0x04

Reset value: 0x0000 0000

Caution: It is mandatory to write CEC_CFGR only when CECEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSTN	OAR[14:0]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFTOP	BRDNOGEN	LBPEGEN	BREGEN	BRESTP	RXTOL	SFT[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LSTN:** Listen mode

LSTN bit is set and cleared by software.

0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.

1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

Bits 30:16 **OAR[14:0]:** Own addresses configuration

The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.

At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN = 1), but without acknowledge sent. Broadcast messages are always received.

Example:

OAR = 0b000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5. Consequently, each message directed to one of these addresses is received.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **SFTOP**: SFT option bit

The SFTOPT bit is set and cleared by software.

0: SFT timer starts when TXSOM is set by software.

1: SFT timer starts automatically at the end of message transmission/reception.

Bit 7 **BRDNOGEN:** Avoid error-bit generation in broadcast

The BRDNOGEN bit is set and cleared by software.

0: BRE detection with BRESTP = 1 and BREGEN = 0 on a broadcast message generates an error-bit on the CEC line. LBPE detection with LBPEGEN = 0 on a broadcast message generates an error-bit on the CEC line.

1: Error-bit is not generated in the same condition as above. An error-bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.

Bit 6 **LBPEGEN:** Generate error-bit on long bit period error

The LBPEGEN bit is set and cleared by software.

0: LBPE detection does not generate an error-bit on the CEC line.

1: LBPE detection generates an error-bit on the CEC line.

Note: If BRDNOGEN = 0, an error-bit is generated upon LBPE detection in broadcast even if LBPEGEN = 0.

Bit 5 **BREGEN:** Generate error-bit on bit rising error

The BREGEN bit is set and cleared by software.

0: BRE detection does not generate an error-bit on the CEC line.

1: BRE detection generates an error-bit on the CEC line (if BRESTP is set).

Note: If BRDNOGEN = 0, an error-bit is generated upon BRE detection with BRESTP = 1 in broadcast even if BREGEN = 0.

Bit 4 **BRESTP**: Rx-stop on bit rising error

The BRESTP bit is set and cleared by software.

0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.

1: BRE detection stops message reception.

Bit 3 **RXTOL:** Rx-tolerance

The RXTOL bit is set and cleared by software.

0: Standard tolerance margin:

- Start-bit, +/- 200 µs rise, +/- 200 µs fall
- Data-bit: +/- 200 µs rise. +/- 350 µs fall

1: Extended tolerance

- Start-bit: +/- 400 µs rise, +/- 400 µs fall
- Data-bit: +/-300 µs rise, +/- 500 µs fall

Bits 2:0 **SFT[2:0]**: Signal free time

SFT bits are set by software. In the SFT = 0x0 configuration, the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.

0x0

- 2.5 data-bit periods if CEC is the last bus initiator with unsuccessful transmission (ARBLST = 1, TXERR = 1, TXUDR = 1 or TXACKE = 1)
- 4 data-bit periods if CEC is the new bus initiator
- 6 data-bit periods if CEC is the last bus initiator with successful transmission (TXEOM = 1)

0x1: 0.5 nominal data bit periods

0x2: 1.5 nominal data bit periods

0x3: 2.5 nominal data bit periods

0x4: 3.5 nominal data bit periods

0x5: 4.5 nominal data bit periods

0x6: 5.5 nominal data bit periods

0x7: 6.5 nominal data bit periods

58.7.3 CEC Tx data register (CEC_TXDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXD[7:0]**: Tx data

TXD is a write-only register containing the data byte to be transmitted.

58.7.4 CEC Rx data register (CEC_RXDR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXD[7:0]**: Rx data

RXD is read-only and contains the last data byte that has been received from the CEC line.

58.7.5 CEC interrupt and status register (CEC_ISR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACKE	TXERR	TXUDR	TXEND	TXBR	ARBLST	RXACKE	LBPE	SBPE	BRE	RXOVR	RXEND	RXBR
			rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKE:** Tx-missing acknowledge error

In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.

TXACKE is cleared by software write at 1.

Bit 11 **TXERR:** Tx-error

In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.

TXERR is cleared by software write at 1.

Bit 10 **TXUDR:** Tx-buffer underrun

In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.

TXUDR is cleared by software write at 1

Bit 9 **TXEND:** End of transmission

TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.

TXEND is cleared by software write at 1.

Bit 8 **TXBR:** Tx-byte request

TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within six nominal data-bit periods before transmission underrun error occurs (TXUDR).

TXBR is cleared by software write at 1.

Bit 7 **ARBLST:** Arbitration lost

ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.

ARBLST is cleared by software write at 1.

Bit 6 **RXACKE:** Rx-missing acknowledge

In receive mode, RXACKE is set by hardware to inform application that no acknowledge was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception.

RXACKE is cleared by software write at 1.

Bit 5 **LBPE:** Rx-long bit period error

LBPE is set by hardware in case a data-bit waveform is detected with long bit period error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an error-bit on the CEC line if LBPEGEN = 1. In case of broadcast, error-bit is generated even in case of LBPEGEN = 0.

LBPE is cleared by software write at 1.

Bit 4 **SBPE:** Rx-short bit period error

SBPE is set by hardware in case a data-bit waveform is detected with short bit period error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an error-bit on the CEC line.

SBPE is cleared by software write at 1.

Bit 3 **BRE***:* Rx-bit rising error

BRE is set by hardware in case a data-bit waveform is detected with bit rising error. BRE is set either at the time the misplaced rising edge occurs, or at the end of the maximum BRE tolerance allowed by RXTOL, in case rising edge is still longing. BRE stops message reception if BRESTP = 1. BRE generates an error-bit on the CEC line if BREGEN = 1.

BRE is cleared by software write at 1.

Bit 2 **RXOVR***:* Rx-overrun

RXOVR is set by hardware if RXBR is not yet cleared at the time a new byte is received on the CEC line and stored into RXD. RXOVR assertion stops message reception so that no acknowledge is sent. In case of broadcast, a negative acknowledge is sent.

RXOVR is cleared by software write at 1.

Bit 1 **RXEND***:* End of reception

RXEND is set by hardware to inform application that the last byte of a CEC message is received from the CEC line and stored into the RXD buffer. RXEND is set at the same time of RXBR.

RXEND is cleared by software write at 1.

Bit 0 **RXBR***:* Rx-byte received

The RXBR bit is set by hardware to inform application that a new byte has been received from the CEC line and stored into the RXD buffer.

RXBR is cleared by software write at 1.

58.7.6 CEC interrupt enable register (CEC_IER)

Address offset: 0x14

Reset value: 0x0000 0000

Caution: It is mandatory to write CEC_IER only when CECEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **TXACKIE***:* Tx-missing acknowledge error interrupt enable

The TXACKEIE bit is set and cleared by software.

0: TXACKE interrupt disabled

1: TXACKE interrupt enabled

Bit 11 **TXERRIE***:* Tx-error interrupt enable

The TXERRIE bit is set and cleared by software.

0: TXERR interrupt disabled

1: TXERR interrupt enabled

Bit 10 **TXUDRIE***:* Tx-underrun interrupt enable

The TXUDRIE bit is set and cleared by software.

0: TXUDR interrupt disabled

1: TXUDR interrupt enabled

Bit 9 **TXENDIE***:* Tx-end of message interrupt enable
The TXENDIE bit is set and cleared by software.
0: TXEND interrupt disabled
1: TXEND interrupt enabled

Bit 8 **TXBRIE***:* Tx-byte request interrupt enable
The TXBRIE bit is set and cleared by software.
0: TXBR interrupt disabled
1: TXBR interrupt enabled

Bit 7 **ARBLSTIE***:* Arbitration lost interrupt enable
The ARBLSTIE bit is set and cleared by software.
0: ARBLST interrupt disabled
1: ARBLST interrupt enabled

Bit 6 **RXACKIE***:* Rx-missing acknowledge error interrupt enable
The RXACKIE bit is set and cleared by software.
0: RXACKE interrupt disabled
1: RXACKE interrupt enabled

Bit 5 **LBPEIE***:* Long bit period error interrupt enable
The LBPEIE bit is set and cleared by software.
0: LBPE interrupt disabled
1: LBPE interrupt enabled

Bit 4 **SBPEIE***:* Short bit period error interrupt enable
The SBPEIE bit is set and cleared by software.
0: SBPE interrupt disabled
1: SBPE interrupt enabled

Bit 3 **BREIE***:* Bit rising error interrupt enable
The BREIE bit is set and cleared by software.
0: BRE interrupt disabled
1: BRE interrupt enabled

Bit 2 **RXOVRIE***:* Rx-buffer overrun interrupt enable
The RXOVRIE bit is set and cleared by software.
0: RXOVR interrupt disabled
1: RXOVR interrupt enabled

Bit 1 **RXENDIE***:* End of reception interrupt enable
The RXENDIE bit is set and cleared by software.
0: RXEND interrupt disabled
1: RXEND interrupt enabled

Bit 0 **RXBRIE***:* Rx-byte received interrupt enable
The RXBRIE bit is set and cleared by software.
0: RXBR interrupt disabled
1: RXBR interrupt enabled

58.7.7 HDMI-CEC register map

Table 708. HDMI-CEC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CEC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXEOM	TXSOM	CECEN
	Reset value																														0	0	0
0x04	CEC_CFGR	LSTN	OAR[14:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	SFTOPT	BRDNOGEN	LBPEGEN	BREGEN	BRESTP	RXTOL	SFT[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x08	CEC_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C	CEC_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXD[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x10	CEC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKE	TXERR	TXUDR	TXEND	TXBR	ARBLST	RXACKE	LBPE	SBPE	BRE	RXOVR	RXEND	RXBR
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	CEC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXACKIE	TXERRIE	TXUDRIE	TXENDIE	TXBRIE	ARBLSTIE	RXACKIE	LBPEIE	SBPEIE	BREIE	RXOVRIE	RXENDIE	RXBRIE
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

59 Debug support (DBG)

59.1 DBG introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Breakpoint debugging of the CPU core
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry-standard debugging tools. A trace port allows data to be captured for logging and analysis.

The debug features are based on Arm® CoreSight™ components.

- SWJ-DP: JTAG/Serial-wire debug port
- AHB-AP: AHB access port
- ROM table
- System control space (SCS)
- Breakpoint unit (BPU)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded Trace Macrocell™ (ETM)
- Cross-trigger interface (CTI)
- Trace port interface unit (TPU)

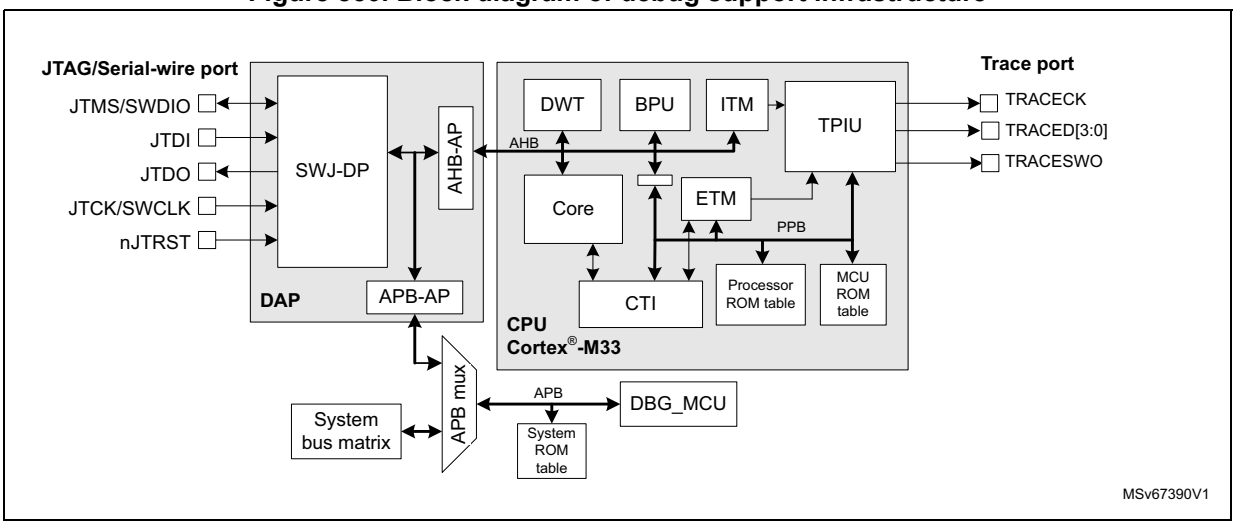
The debug features are accessible by the debugger via the AHB-AP.

Additional information can be found in the Arm® documents referenced in *Section 59.13*.

59.2 DBG functional description

59.2.1 DBG block diagram

Figure 850. Block diagram of debug support infrastructure



59.2.2 DBG pins and internal signals

Table 709. JTAG/Serial-wire debug port pins

Pin name	JTAG debug port		SW debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	IO	Serial-wire data in/out	PA13
JTCK/SWCLK	I	JTAG test clock	I	Serial-wire clock	PA14
JTDI(1)	I	JTAG test data input	-	-	PA15
JTDO	O	JTAG test data output	-	-	PB3
nJTRST	I	JTAG test reset	-	-	PB4

1. TDI is hosted on the same I/O as a USBPD-CC line. To avoid pull-up/down conflict, a user option can help to decide whether the pad is used as TDI or as CC.

Table 710. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED0	O	Trace synchronous data out 0	Refer to the datasheet
TRACED1		Trace synchronous data out 1	
TRACED2		Trace synchronous data out 2	
TRACED3		Trace synchronous data out 3	
TRACECK		Trace clock	

Table 711. Single-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Single-wire trace asynchronous data out	PB3(1)

1. TRACESWO is multiplexed with JTDO. This means that a single-wire trace is only available when using the serial-wire debug interface, and not when using JTAG.

59.2.3 DBG reset and clocks

The debug port (SWJ-DP) is reset by a power-on reset and when waking up from Standby mode.

The debugger supplies the clock for the debug port via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG modes, as well as to operate the state machines and internal logic of the debug port. This clock must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the DCLK domain that covers the rest of the SWJ-DP and the access port.

The DCLK is a gated version of the system clock.

The DCLK domain is enabled by the debugger using the CDBGPWRUPREQ bit in the *DP control and status register (DP_CTRLSTATR)*. The clock must be enabled before the debugger can access any of the debug features on the device. The availability of the clock is reflected in the CDBGPWRUPACK bit in DP_CTRL/STATR. The DCLK is disabled at power-up, and must be disabled when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor are clocked with the processor clock.

59.2.4 DBG power domains

The debug components are located in the core power domain. This means that the debugger connection is not possible in Shutdown or Standby low-power mode. To avoid losing the connection when the device enters Standby mode, the power can be maintained to the core by setting a bit in the *DBGMCU configuration register (DBGMCU_CR)*. This also keeps the processor clocks active and holds off the reset, so that the debug session is maintained.

59.2.5 Debug and low-power modes

The devices include power-saving features that allow the core power domain to be switched off or stopped when not required. If the power is switched off or if the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, power-saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power-saving mode, but the debugger does not lose the connection.

The emulation mode is programmed in the microcontroller debug (DBGMCU) unit. For more information, refer to *Section 59.12*.

59.2.6 Security

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope. For example, secure software debug and trace can be disabled without preventing the debug of nonsecure code.

Debugger access to secure memory (when permitted) must be performed using secure transactions on the debug AHB, that is, with the PROT[6] bit set in the *AP1 control/status word register (AP1_CSWR)*.

Debugger access is disabled while the processor is booting from system flash memory.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features

  0: All debug features are disabled.

  1: Debug features in the nonsecure state are enabled. Debug features in the secure state are dependent on the state of the **spiden** signal.
- **spiden**: enables debug in the secure state when **dbgen** = 1.

  0: Debug features are disabled in the secure state.

  1: Debug features are enabled in the secure state.
- **niden**: enables trace and performance monitoring (noninvasive debug).

  0: Trace generation is disabled.

  1: Trace generation in the nonsecure state is enabled. Trace generation in the secure state is dependent on the state of the **spniden** signal.
- **spniden**: enables trace and performance monitoring in the secure state when **niden** = 1.

  0: Trace generation is disabled in the secure state.

  1: Trace generation is enabled in the secure state.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant chapter, or to the relevant Arm® technical documentation.

The state of the signals is set according to the debug state as shown in *Table 712*, when TrustZone® is enabled (TZEN = 0xB4), and *Table 713* when TrustZone is disabled (TZEN = 0xC3).

Table 712. Authentication signal states with TrustZone enabled (TZEN = 0xB4)

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 spiden = 1 niden = 1 spniden = 1	Debug and trace is enabled whatever the state of the processor. Debugger access to secure memory is permitted.
CLOSED SECURE	dbgen = 1 spiden = 0 niden = 1 spniden = 0	Debug and trace is enabled when the processor is in nonsecure state. Debugger access to nonsecure memory is permitted. Debugger access to secure memory is disabled.
CLOSED	dbgen = 0 spiden = 0 niden = 0 spniden = 0	Debug and trace is disabled.

Table 713. Authentication signal states with TrustZone disabled (TZEN = 0xC3)

Debug state	Authentication signal state	Description
OPEN	dbgen = 1 spiden = 1 niden = 1 spniden = 1	Debug and trace is enabled whatever the state of the processor. All memory and resources are considered nonsecure and accessible to the debugger.
CLOSED	dbgen = 0 spiden = 0 niden = 0 spniden = 0	Debug and trace is disabled.

The state of the authentication signals can be read from the DAUTHSTATUS register in the system control space (SCS) of the Cortex®-M33.

The debug state depends upon the product life cycle state and the debug authentication state (see *Section 59.2.7*).

Table 714. Life cycle state and debug states

Product life cycle state	Debug state
OPEN	OPEN
TZ_CLOSED	CLOSED-SECURE
CLOSED (debug not authenticated)	CLOSED

Table 714. Life cycle state and debug states

Product life cycle state	Debug state
CLOSED (debug constrained)	OPEN/CLOSED-SECURE(1)
LOCKED	CLOSED

1. Depends on authorization level.

59.2.7 Debug authentication

Figure 851. Product life cycle states and debug authentication

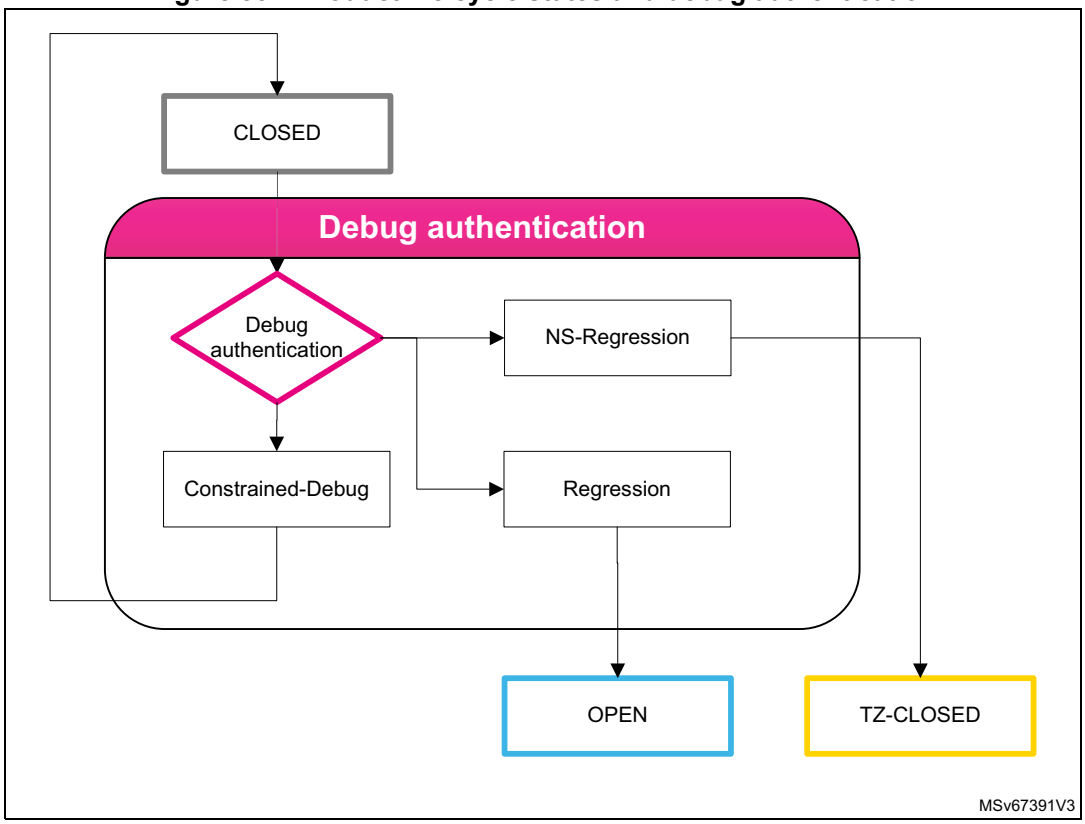


Figure 851 shows the product life cycle and debug authentication states.

If the device is in CLOSED (product life cycle) state, the debug state is CLOSED. The debug authentication procedure allows a trusted debugger to reopen access without compromising sensitive information called the Root Of Trust (ROT).

Reopening the debug is possible only if sensitive asset security is ensured (and TrustZone is enabled (TZEN = 0xB4). This is called Constrained Debug, as constraints ensure the security of the ROT information.

Alternatively, a partial or full regression mechanism can be used when security of sensitive information cannot be guaranteed. This is called Regression, as regression ensures removal of the sensitive information before reopening the debug.

- Partial regression corresponds to releasing nonsecure code and assets. The intermediate state which allows partial regression management is called NS-Regression.
- Full regression corresponds to releasing all code and assets. The intermediate state allowing full regression management is called Regression.

Debug authentication control principle

The debug authentication is one of the most critical security features of the system considering that with a debugger the user can access a large part of the system.

To control the reopening of the debug, the device imposes a debug authentication protocol.

The protocol implements a challenge response mechanism based on asymmetric cryptography to authenticate the host. It relies on a key pair, with a Public Key stored in the device, and a Private Key from the host library, used to sign a random value (the challenge) generated by the device.

The protocol implements a bidirectional communication between the host and the device through a mailbox interface located in the DBGMCU.

The host can write to the mailbox via the JTAG/SWD interface. It expects to get responses and messages from the device via the same mailbox.

The debug authentication protocol is launched on a power-on reset of the device, when an “open request” message is posted by the host.

The protocol is based on:

- Initial message: posted by the host combined with a reset to launch the debug authentication process on the device.
- Challenge message: the device generates a random value, to be signed by the host, when sending back the response.
- Response: the host sends a message to the device proving its authenticity. This is done using a tool to generate a token.

The implementation is ensured by code embedded in the system flash. This code is called automatically after reset if an initial message has been posted by the host in the mailbox.

After a first sequence of mutual authentication to align on protocol version, type of device, and similar parameters, the device generates a random value that to be signed by the host with a private key when building the response.

The STMicroelectronics implementation is based on the Arm® PSA-ADAC solution for debug authentication.

The debug authentication can be implemented using a proprietary or open-source protocol.

As this feature is critical for security, the STM32H5 devices come with debug authentication provisioned in system flash. STMicroelectronics provides the host tools integrated with some debug environments (IDEs).

Debug authentication provisioning

Debug authentication is natively supported by the STM32H5 platform. The data used by ST debug authentication (ST-DA) must be provisioned at a defined location in the secure key storage area (OBKeys defined in flash memory).

The debug authentication configuration must be done only when the PRODUCT_STATE is “Provisioning”, cannot be performed when PRODUCT_STATE is “Open”.

The following data must be provisioned (refer to *Section 7.7.2: RSS user functions*):

- Data must be provisioned at the very beginning of the OBK_HDPL1 mapping (0x0FFD 0100) (see *Section 7.5.2: OBK access per HDPL level*)
- RSS_Lib encryption option must be set for STM32H533/573xx devices
- RSS_Lib encryption option must be reset for STM32H523/56xxx devices

The data to provision are defined in *Table 715*. They depend upon the TZEN option byte setting (enabled or disabled).

Table 715. Definition of data to provision

Data size	TZEN enabled (certificates)	TZEN disabled (password: only full regression is allowed)
32 bytes	SHA256 of the overall blob (i.e. the two next fields)	
	SHA256 of the ROT certificate public key	SHA256 of the password
16 bytes	128-bis permission mask (see *Table 716*)	128 bits are reserved, to be set to 0

Caution: Configure debug authentication as explained above before changing to states different from Open or Provisioning.

Table 716. Permission mask (Endianness: Little Endian)

Bit(s)	Description
Bit 0	When set it allows Intrusive debug from HDPL1 – NS
Bit 1	When set it allows Intrusive debug from HDPL2 – NS
Bit 2	When set it allows Intrusive debug from HDPL3 – NS
Bit 3	Reserved
Bit 4	When set it allows Intrusive debug from HDPL1 – S
Bit 5	When set it allows Intrusive debug from HDPL2 – S
Bit 6	When set it allows Intrusive debug from HDPL3 – S
Bits 7:11	Reserved
Bit 12	Regression to TZ-Closed, when 1 regression to TZ-Closed is allowed
Bit 13	Reserved
Bit 14	Regression (Full regression), when 1 full regression is allowed
Bit 15:127	Reserved

59.3 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- A 5-pin standard JTAG interface (JTAG-DP)
- A 2-pin (clock + data) serial-wire debug port (SW-DP)

These two modes are mutually exclusive, since they share the same I/O pins.

By default, the JTAG-DP is selected after a system or a power-on reset. The five I/O pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and nJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

... (50 or more ones) ..., 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, ... (50 or more ones) ...

JTCK/SWCLK must be cycled for each data bit.

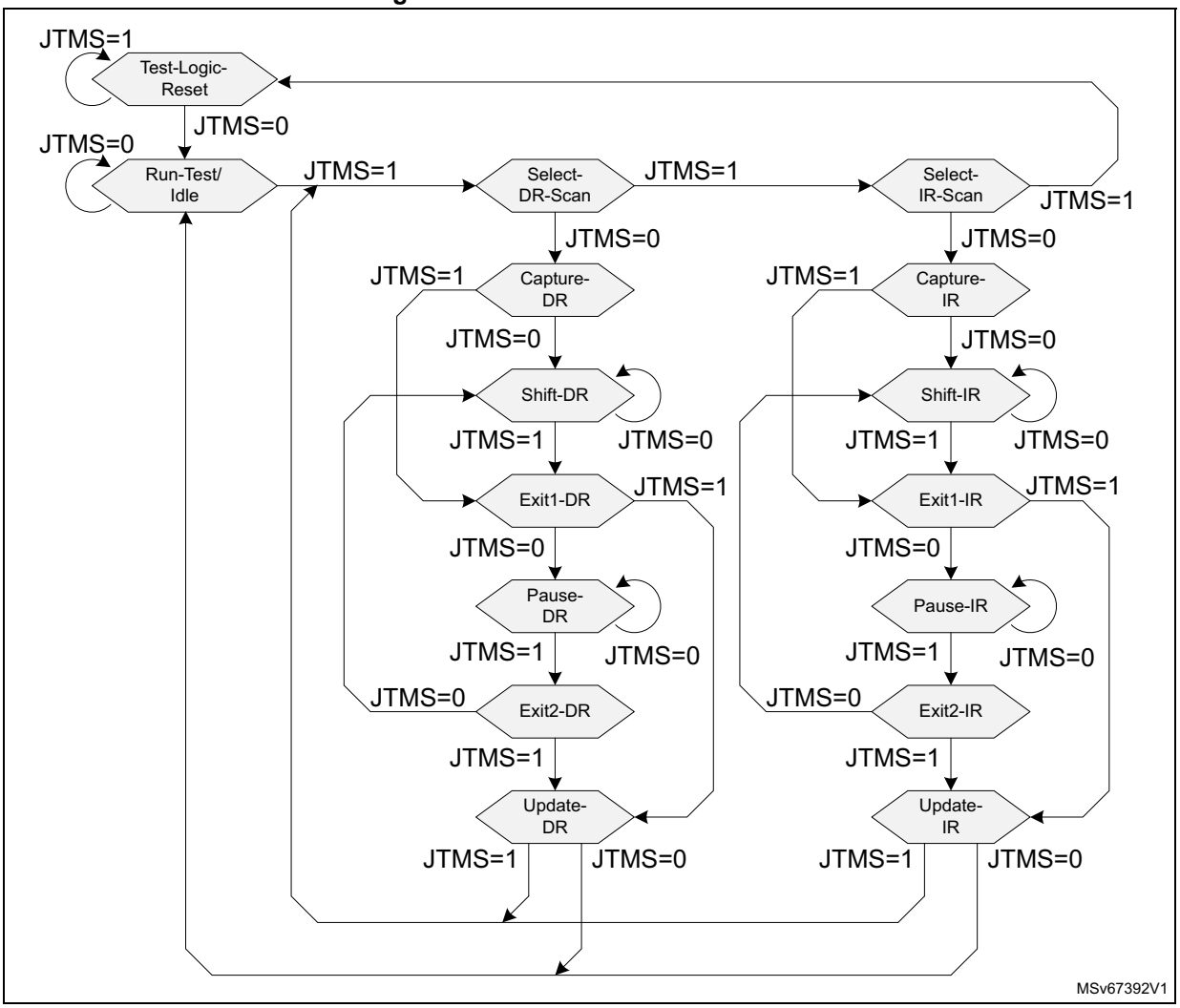
In SW-DP mode, the unused JTAG pins JTDI, JTDO, and nJTRST can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software, but debugging is no longer possible.

59.3.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM), shown in the figure below, based on IEEE Std 1149.1-1990. The state machine controls two scan chains, one associated with an instruction register (IR) and the other one with a number of data registers (DR).

Figure 852. JTAG TAP state machine



The operation of the JTAG-DP is as follows:

1. When the TAPSM goes through the Capture-IR state, 0b0001 is transferred to the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.
2. While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
   – The LSB of the IR scan chain is output on JTDO.
   – Bit[n] of the IR scan chain is transferred to bit[n-1].
   – The value on JTDI is transferred to the MSB of the IR scan chain.
3. When the TAPSM goes through the Update-IR state, the value scanned into the IR scan chain is transferred to the instruction register.
4. When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers to one of the DR scan chains, connected between JTDI and JTDO.
5. The value held in the instruction register determines which data register, and associated DR scan chain, are selected.
6. This data is then shifted while the TAPSM is in the Shift-DR state, in the same manner as the IR shifts in the Shift-IR state.
7. When the TAPSM goes through the Update-DR state, the value scanned into the DR scan chain is transferred to the selected data register.
8. When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the test-logic-reset state.

The data registers corresponding to the 4-bit IR instructions are listed in the table below.

Table 717. JTAG-DP data registers

IR instruction	DR register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	ABORT register – bits 34:3 = APABORT[31:0]: Write 0x0000 0001 to abort an ongoing access port transaction – bits 2:1 = A[3:2] = 00 – bit 0 = RnW = 0: write only
1001	(BYPASS)	1	Reserved: BYPASS selected

Table 717. JTAG-DP data registers (continued)

IR instruction	DR register	Scan chain length	Description
1010	DPACC	35	Debug port access register Initiates the debug port and gives access to a debug port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request bits 2:1 = A[3:2] = 2-bit address of a debug port register bit 0 = RnW = read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: – 010 = OK/FAULT – 001 = WAIT – others = reserved
1011	APACC	35	Access port access register Initiates an access port and gives access to an access port register. – When transferring data IN: bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request bits 2:1 = A[3:2] = 2-bit sub-address of an access port register bit 0 = RnW= Read request (1) or write request (0) – When transferring data OUT: bits 34:3 = DATA[31:0] = 32-bit data read following a read request bits 2:0 = ACK[2:0] = 3-bit Acknowledge: – 010 = OK/FAULT – 001 = WAIT – others= reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected
1110	IDCODE	32	Identification code 0x6BA0 0477: Cortex®-M33 JTAG debug port ID code
1111	BYPASS	1	Bypass A single JTCK cycle delay is inserted between JTDI and JTDO.

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification (see *Section 59.13*).

59.3.2 Serial-wire debug port

The serial-wire debug protocol uses the following pins:

- SWCLK: clock from host to target
- SWDIO: bidirectional serial data (100 kΩ pull-up required)

Serial data is transferred LSB first, synchronously with the clock.

A transfer comprises three phases:

1. Packet request (8 bits) transmitted by the host, see *Table 718*
2. Acknowledge response (3 bits) transmitted by the target, see *Table 719*
3. Data transfer (33 bits) transmitted by the host (in case of a write) or target (in case of a read), see *Table 720*

The data transfer only occurs if the acknowledge response is OK.

Between each phase, if the direction of the data is reversed, a single clock-cycle turn-around time is inserted.

Table 718. Packet request

Bit field	Name	Description
0	Start	Must be 1
1	APnDP	– 0: DP register access - see *Section 59.3.3* – 1: AP register access - see *Section 59.4*
2	RnW	– 0: write request – 1: read request
4:3	A(3:2)	Address field of the DP or AP register (refer to *Table 722* or *Table 724*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host, must be read as 1 by target

Table 719. ACK response

Bit field	Name	Description
2:0	ACK	– 000: FAULT – 010: WAIT – 100: OK

Table 720. Data transfer

Bit field	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single-bit parity of 32 data bits

In the case of a FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled: in this case, the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists in 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the serial-wire debug protocol, refer to the Arm® Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

59.3.3 Debug port registers

Both the SW-DP and the JTAG-DP access the debug port (DP) registers listed in *Table 722*.

The debugger can access the DP registers as follows:

1. Program the A(3:2) field in the DPACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the data field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 718*). The write data are sent in the data phase.
2. To access one of the banked DP registers at the address 0x4, the register number must first be written to the DP_SELECTR register at the address 0x8. Any subsequent read or write to address 0x4 accesses the register corresponding to the contents of the DP_SELECTR register.

Table 721. Debug port registers

Address	A(3:2) value	R/W	Description
0x0	00	R	*DP debug port identification register (DP_DPIDR)* contains the IDCODE for the debug port.
		W	*DP abort register (DP_ABORTR)*(1) aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRLSTATR register.
0x4	01	R/W	If DP_SELECTR.DPBANKSEL[3:0] = 0x0, *DP control and status register (DP_CTRLSTATR)* controls the DP and provides status information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x1, *DP data link control register (DP_DLCR)*(2) controls the operating mode of the SWD data link.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x2, *DP target identification register (DP_TARGETIDR)* provides target identification information.
			If DP_SELECTR.DPBANKSEL[3:0] = 0x3, *DP data link protocol identification register (DP_DLPIDR)*(2) provides the SWD protocol version.
0x8	10	R	*DP event status register (DP_RESENDR)*(2) returns the value that was returned by the last AP read or DP_RDBUFF read. Used in the event of a corrupted read transfer.
		W	*DP access port select register (DP_SELECTR)* selects the access port, access port register bank, and DP register at the address 0x4.
0xC	11	R	*DP read buffer register (DP_RDBUFFR)* – Via JTAG-DP, this register allows the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation) – Via SW-DP, this register contains the result of the preceding AP read access, allowing a new AP access to be avoided.

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

DP debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: revision code
0x6: Rev 6

Bits 27:20 **PARTNO[7:0]**: part number for the debug port
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: minimal debug port (MINDP) implementation
0x0: MINDP not implemented

Bits 15:12 **VERSION[3:0]**: debug port architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm® JEDEC code

Bit 0 Reserved, must be kept at reset value.

DP abort register (DP_ABORTR)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
											w	w	w		w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: overrun error clear
0: no effect
1: STICKYORUN bit cleared in DP_CTRL/STATR register

Bit 3 **WDERRCLR**: write data error clear
0: no effect
1: WDATAERR bit cleared in DP_CTRL/STATR register

Bit 2 **STKERRCLR**: sticky error clear
0: no effect
1: STICKYERR bit cleared in DP_CTRL/STATR register

Bit 1 Reserved, must be kept at reset value.

Bit 0 **DAPABORT**: current AP transaction aborted if excessive number of WAIT responses returned
This bit indicates that the transaction is stalled.
0: no effect
1: transaction aborted

DP control and status register (DP_CTRLSTATR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
								r	r	r				r	r

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **CDBGPWRUPACK**: debug power-up acknowledge
See description in *Section 59.2.5: Debug and low-power modes*.
0: DCLK gated
1: DCLK enabled

Bit 28 **CDBGPWRUPREQ**: debug power-up request
This bit controls the DCLK enable request signal.
0: requests DCLK gating
1: requests DCLK enable

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **WDATAERR**: write data error (read-only) in SW-DP

This bit indicates that there is a parity or framing error on the data phase of a write, or a write accepted by the DP is then discarded without being submitted to the AP.

This bit is reset by writing 1 to the ABORT.WDERRCLR bit.

0: no error

1: an error occurred

Note: This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response (read-only) in SW-DP

This bit indicates the response to the last AP read access.

0: read not OK

1: read OK

Note: This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: transaction error (read-only in SW-DP, read/write in JTAG-DP)

This bit indicates that an error occurred in an AP transaction.It is reset by writing 1 to the DP_ABORTR.STKERRCLR bit (in SW-DP and JTAG-DP)

0: no error

1: an error occurred

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **STICKYORUN**: overrun (read-only in SW-DP, read/write in JTAG-DP).

This bit indicates that an overrun occurred (new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set. it is reset by writing 1 to the DP_ABORTR.ORUNERRCLR bit (in SW-DP and JTAG-DP).

0: no overrun

1: an overrun occurred

Bit 0 **ORUNDETECT**: overrun detection mode enable.

0: disabled

1: enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.

DP data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
						r	r	r	r						

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: tristate period for SWDIO
0x0: 1 data bit period

Bits 7:6 **WIREMODE[1:0]**: SW-DP mode
0x0: synchronous mode

Bits 5:0 Reserved, must be kept at reset value.

DP target identification register (DP_TARGETIDR)

Address offset: 0x4

Reset value: 0xXXXX 0041

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: target revision

Bits 27:12 **TPARTNO[15:0]**: target part number
0x4840: STM32H562/563/573
0x4780: STM32H523/533

Bits 11:1 **TDESIGNER[10:0]**: target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

DP data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

This register is accessible when DP_SELECTR.DPBANKSEL[3:0] = 0x3.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: target instance number
this field defines the instance number for the device in a multi-drop system.
0x0: instance number 0

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial-wire debug protocol version
0x1: version 2

DP event status register (DP_RESENDR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: value returned by the last AP read or DP_RDBUFF read
This register is used in the event of a corrupted read transfer.

DP access port select register (DP_SELECTR)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: access port select
This field selects the access port for the next transaction.
0x00: AP0 - System debug access port (APB-AP)
0x01: AP1 - Cortex®-M33 debug access port (AHB-AP)
Others: reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select
This field selects the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select

This field selects the register at the address 0x4 of the debug port.

0x0: DP_CTRLSTAT register

0x1: DP_DLCR register

0x2: DP_TARGETID register

0x3: DP_DLPIDR register

Others: reserved

DP read buffer register (DP_RDBUFFR)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: value returned by the last AP read access

The value returned by an AP read access can be obtained using a second read access to the same address (initiates a new transaction on the corresponding bus), or can be read from this register, in which case no new AP transaction occurs.

59.3.4 Debug port register map and reset values

These registers are not on the CPU memory bus. They are accessed only through the SW-DP and JTAG-DP debug interfaces.

The debug port address offset is 4-bit wide, where the two most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 722. Debug port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	DP_DPIDR	REVISION [3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION [3:0]				DESIGNER[10:0]											Res.
	Reset value	0	1	1	0	1	0	1	1	1	0	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	1
0x0	DP_ABORTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	Res.	DAPABORT
	Reset value																												0	0	0		0

Table 722. Debug port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4(1)	DP_CTRLSTATR	Res.	Res.	CDBGPWRUPACK	CDBGPWRUPREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDATAERR	READOK	STICKYERR	Res.	Res.	Res.	STICKYORUN	ORUNDETECT
	Reset value			0	0																					0	0	0				0	0
0x4(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		WIREMODE[1:0]		Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																							0	0	0	0						
0x4(3)	DP_TARGETIDR	TREVISION [3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	
0x4(4)	DP_DLPIDR	TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x8	DP_RESENDR	RESEND[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	DP_SELECTR	APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
	Reset value	x	x	x	x	x	x	x	x																	x	x	x	x	x	x	x	x
0xC	DP_RDBUFFR	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. DP_SELECTR.DPBANKSEL[3:0] = 0x0.
2. DP_SELECTR.DPBANKSEL[3:0] = 0x1.
3. DP_SELECTR.DPBANKSEL[3:0] = 0x2.
4. DP_SELECTR.DPBANKSEL[3:0] = 0x3.

59.4 Access ports

There are two access ports (AP) attached to the DP.

- System debug access port (AP0): Enables access to the DBGMCU and the system ROM table via an APB bus.
- Cortex®-M33 debug access port (AP1): Enables access to the debug and trace features integrated in the Cortex®-M33 processor core via its internal AHB bus.

59.4.1 Access port registers

The access ports are of type MEM-AP: the debug and trace component registers are mapped in the address space of the AHB. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in *Table 724*.

The address of the AP registers is composed of the following fields:

- bits [7:4]: content of the APBANKSEL[3:0] field in the *DP access port select register (DP_SELECTR)*)
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 717*), or of the SW-DP packet request (see *Table 718*), depending on the debug interface used
- bits [1:0]: always set to 0

The content of the DP_SELECTR.APSEL[3:0] field defines which MEM-AP is being accessed.

Table 723. MEM-AP registers

Address	APBANKSEL	A(3:2)	Name	Description
0x00	0x0	0	CSWR	Control/status word register
0x04	0x0	1	TAR	Transfer address register Target address for the bus transaction.
0x08	-	-	-	Reserved
0x0C	0x0	3	DRWR	Data read/write register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0]
0x10	0x1	0	BD0R	Banked data 0 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x0.
0x14	0x1	1	BD1R	Banked data 1 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x4.
0x18	0x1	2	BD2R	Banked data 2 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0x8.
0x1C	0x1	3	BD3R	Banked data 3 register Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] + 0xC.
0x20	-	-	-	Reserved
0x24 to 0xEC	-	-	-	Reserved
0xF0	-	-	-	Reserved
0xF4	0xF	1	CFGR	Configuration register (read only)
0xF8	0xF	2	BASER	Debug base address register (read only) Base address of the ROM table

Table 723. MEM-AP registers (continued)

Address	APBANKSEL	A(3:2)	Name	Description
0xFC	0xF	3	IDR	Identification register (read only)

The debugger can access the AP registers as follows:

1. Program the APSEL[3:0] field in the *DP access port select register (DP_SELECTR)* to choose the AP, and the APBANKSEL[3:0] field in DP_SELECTR to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC data register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 718*). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the AP registers (using the above AP register access procedure) as follows:

1. Program the transaction target address in the *APx transfer address register (APx_TAR) (x = 0, 1)*.
2. Program the *AP1 control/status word register (AP1_CSWR)*, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the *APx data read/write register (APx_DRWR) (x = 0, 1)* to initiate a bus transaction at the address held in AP_TAR. Alternatively, a read or write to the *APx banked data n register (APx_BDnR) (x = 0, 1)* triggers access to the TAR[31:4] + n address, allowing up to four consecutive addresses to be accessed without changing the address in the AP_TAR register.

For more detailed information on the MEM-AP refer to the Arm® Debug Interface Architecture Specification *[1]*.

AP0 control/status word register (AP0_CSWR)

Address offset: 0x0

Reset value: 0x8000 0042

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBGSWEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		r	r	r

Bit 31 **DBGSWEN**: software access enable
Enables or disables software access to the APB bus
0: disable software access
1: enable software access

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: mode of operation
0b0000: normal download or upload
other: reserved

Bit 7 **TRINPROG**: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: no APB transfer in progress
1: APB transfer in progress

Bit 6 **DEVICEEN**: device enable status (read only)
1: APB transfers always enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size in bytes of the transaction (SIZE field)
other: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x2: word (32-bit)

AP1 control/status word register (AP1_CSWR)

Address offset: 0x0

Reset value: 0x43X0 00X2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw	rw	rw	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
								r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: secure transfer request
This field sets the protection attribute HPROT[6] of the bus transfer.
0: secure transfer, HPROT[6] = low
1: nonsecure transfer, HPROT[6] = high
If SPIDEN = 0 and SPROT = 0, no bus transfer occurs

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:24 **PROT[3:0]**: bus transfer protection
This field sets the protection attributes HPROT[3:0] of the bus transfer.
0bXXX0: instruction access
0bXXX1: data access
0bXX0X: user mode
0bXX1X: privilege mode
0bX0XX: non-bufferable
0bX1XX: bufferable
0b0XXX: non-shareable, no look-up, non-modifiable
0b1XXX: shareable, look-up, modifiable

Bit 23 **SPISTATUS**: secure debug authentication status
This field indicates the state of the SPIDEN signal
0: No secure AHB transfers allowed
1: Secure AHB transfers allowed

Bits 22:8 Reserved, must be kept at reset value.

Bit 7 **TRINPROG**: transfer in progress (read only)
This field indicates whether a transfer is currently in progress on the APB master port
0: No AHB transfer in progress
1: AHB transfer in progress

Bit 6 **DBGSTATUS**: debug enable (DBGEN) status
0: AHB transfers blocked
1: AHB transfers enabled

Bits 5:4 **ADDRINC[1:0]**: auto-increment mode
Defines whether TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: address incremented by the size of the transaction (SIZE field), in bytes. Single transfer.
0x2: address incremented by the size of the transaction (SIZE field), in bytes. Packs four 8-bit or two 16-bit transfers into a 32-bit DAP transfer. Multiple transactions are carried out on the AHB interface.
Others: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: size of next memory access transaction
0x0: byte (8-bit)
0x1: halfword (16-bit)
0x2: word (32-bit)
Others: reserved

APx transfer address register (APx_TAR) (x = 0, 1)

Address offset: 0x04

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TA[31:0]**: address of current transfer. In AP0, TA[1:0] are fixed at 0.

APx data read/write register (APx_DRWR) (x = 0, 1)

Address offset: 0x0C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TD[31:0]**: data of current transfer

APx banked data n register (APx_BDnR) (x = 0, 1)

Address offset: 0x10 + 0x4 * n, (n = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]:** banked data of current transfer to address TA [31:4] + 4 * n.
Auto address incrementing is not performed on APx_BD0-3R. Banked transfers are only supported for word transfers.

APx base address register (APx_BASER) (x = 0, 1)

Address offset: 0xF8

Reset value: AP 0: 0xE00E 0003

Reset value: AP 1: 0xE00F E003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[31:12]**: base address (bits 31 to 12) of the first ROM table
The 12 LSBs are zero since the ROM table must be aligned on a 4-Kbyte boundary.
0xE00E0: AP0
0xE00FE: AP1

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: base-address register format
1: Arm® debug interface v5

Bit 0 **ENTRYPRESENT**: debug components presence
Indicates that debug components are present on the access port bus.
1: debug components present

APx identification register (APx_IDR) (x = 0, 1)

Address offset: 0xFC

Reset value: AP 0: 0x5477 0002

Reset value: AP 1: 0x8477 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3]
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLASS[2:0]			Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
r	r	r						r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: revision number
0x5: r1p0
0x8: r0p9

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm®

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm®

Bits 16:13 **CLASS[3:0]**:
0x1: MEM-AP

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**:
0x1: AHB-AP
0x2: APB-AP

59.4.2 Access port register map

These registers are not on the CPU memory bus. They are only accessed through SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by DP_SELECTR.APBANKSEL[3:0] field and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 724. Access port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP0_CSWR	DBGSWEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value	1																				0	0	0	0	0	1	0	0		0	1	0
0x00	AP1_CSWR	Res.	SPROT	Res.	Res.	PROT[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		1			0	0	1	1	X												0	0	0	0	0	X	0	0		0	1	0
0x04	APx_TAR	TA[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x08	Reserved	Reserved																															
0x0C	APx_DRWR	TD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x10	APx_BD0R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	APx_BD1R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	APx_BD2R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	APx_BD3R	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20 to 0xF4	Reserved																																
0xF8	AP0_BASER	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0											1	1

Table 724. Access port register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF8	AP1_BASER	BASEADDR[31:12]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0											1	1
0xFC	AP0_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	1	0
0xFC	AP0_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							CLASS[3:0]				Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0						0	0	0	0	0	0	0	1

59.5 ROM tables

The ROM table is a CoreSight component that contains the base addresses of the CoreSight debug components accessible via the access port to which it is attached. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There is one top-level ROM table behind each access port, APn. The base address of this ROM table can be obtained by reading the APn_BASER register of the access port. The top-level ROM table may point in turn to other ROM tables.

The system ROM table is pointed to by the AP0 base register, AP0_BASER. It contains the base address pointer for the DBGMCU.

The system ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00E 4000 to 0xE00E 4FFC, when accessed by the debugger. It can be accessed by the CPU at the address range 0x4402 4000 to 0x4402 4FFC.

Table 725. System ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	DBGMCU	0xE00E 4000 (debugger) 0x4402 4000 (CPU)	0x0000 4000	4	0x0000 4003
0x004	Top of table	-	-	-	0x0000 0000
0x008 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 728*

There are two ROM tables in the CPU subsystem. The MCU ROM table is pointed to by the AP1 base register, AP1_BASER. It contains the base-address pointer for the processor ROM table and for the TPIU registers.

The MCU ROM table (see the table below) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFC.

Table 726. MCU ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	Processor ROM table	0xE00F F000	0x0000 1000	4	0x0000 1003
0x004	TPIU	0xE004 0000	0xFFF4 2000	4	0xFFF4 2003
0x008	Reserved	-	-	-	0x1FF0 2002
0x00C	Reserved	-	-	-	0x1FF0 2002
0x010	Top of table	-	-	-	0x0000 0000
0x014 to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	See *Table 729*

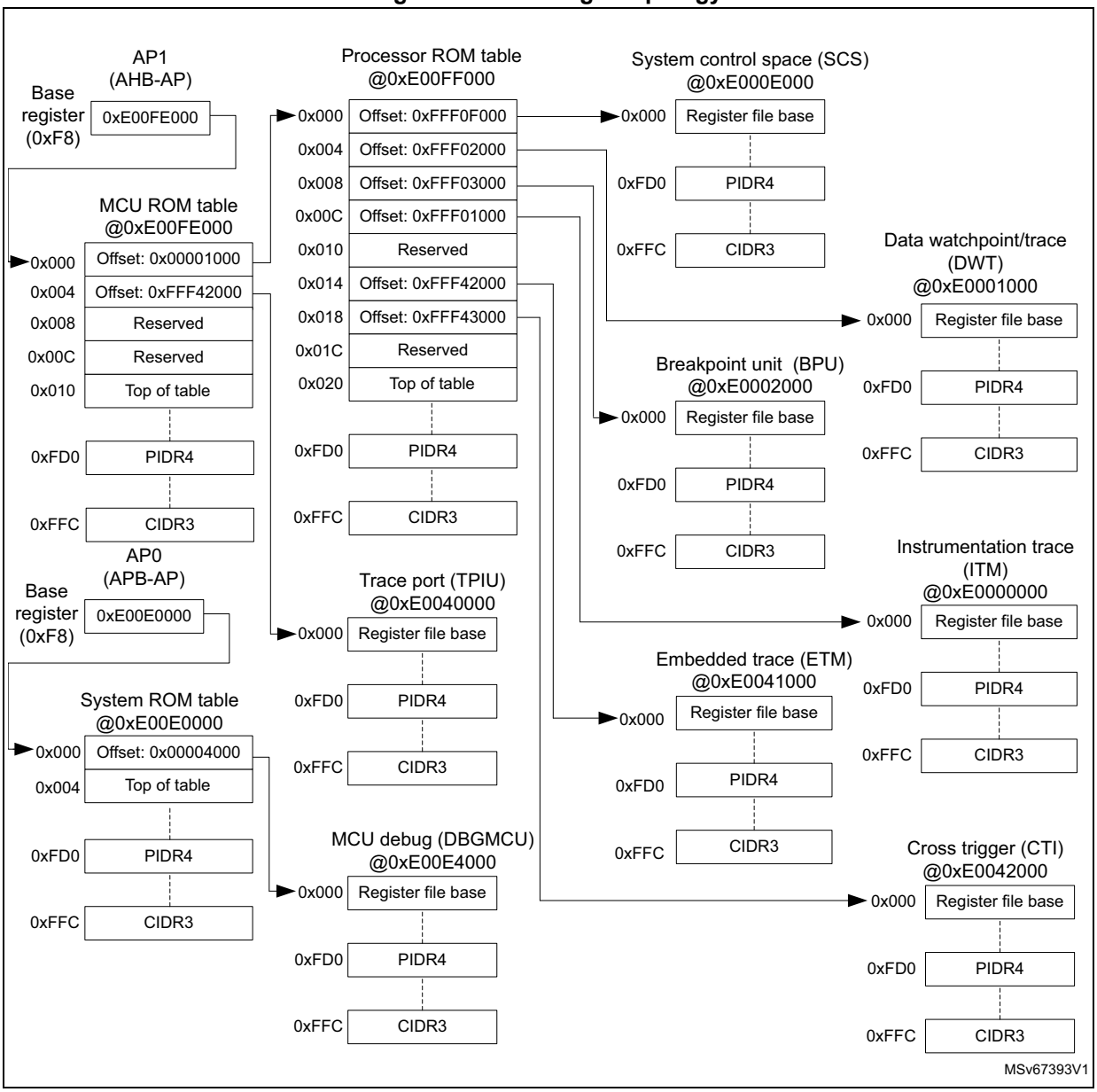
The processor ROM table contains the base-address pointer for the system control space (SCS) registers, that allow the debugger to identify the CPU core, as well as for the BPU, DWT, ITM, ETM, and CTI.

The processor ROM table (see *Table 727*) occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F F000 to 0xE00F FFFC.

Table 727. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000	4	0xFFF0 2003
0xE00F F008	BPU	0xE000 2000	0xFFF0 3000	4	0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000	4	0xFFF0 1003
0xE00F F010	Reserved	-	-	-	0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000	4	0xFFF4 2003
0xE00F F018	CTI	0xE004 2000	0xFFF4 3000	4	0xFFF4 3003
0xE00F F01C	Reserved	-	-	-	0xFFF4 4002
0xE00F F020	Top of table	-	-	-	0x0000 0000
0xE00F F024 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00F FFCC to 0xE00F FFFC	ROM table registers	-	-	-	See *Table 730*

Figure 853. CoreSight topology



59.5.1 System ROM table registers

System ROM memory type register (SYSROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
0x1: system memory present on this bus

System ROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

System ROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x84: STM32H562/563/573
0x78: STM32H523/533

System ROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: STM32H5 series

System ROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

System ROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

System ROM CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

System ROM CoreSight peripheral identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

System ROM CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

System ROM CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

59.5.2 System ROM table register map

Table 728. System ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	SYSROM_ MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to FDC	Reserved																																
0xFE0	SYSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0xFE4	SYSROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	X	X	X	X
0xFE8	SYSROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	SYSROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SYSROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SYSROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	SYSROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SYSROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 725* for register boundary addresses.

59.5.3 MCU ROM table registers

MCU ROM memory type register (MCUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory
0x1: system memory present on this bus

MCU ROM CoreSight peripheral identity register 4 (MCUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

MCU ROM CoreSight peripheral identity register 0 (MCUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x84: STM32H562/563/573
0x78: STM32H523/533

MCU ROM CoreSight peripheral identity register 1 (MCUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 000X

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: STM32H5 series

MCU ROM CoreSight peripheral identity register 2 (MCUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

MCU ROM CoreSight peripheral identity register 3 (MCUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

MCU ROM CoreSight component identity register 0 (MCUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

MCU ROM CoreSight peripheral identity register 1 (MCUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

MCU ROM CoreSight component identity register 2 (MCUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

MCU ROM CoreSight component identity register 3 (MCUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

59.5.4 MCU ROM table register map

Table 729. MCU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	MCUROM_ MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	MCUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to FDC	Reserved																																

Table 729. MCU ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	MCUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0xFE4	MCUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	X	X	X	X
0xFE8	MCUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	MCUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	MCUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	MCUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	MCUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	MCUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 726* for register boundary addresses.

59.5.5 Processor ROM table registers

CPU ROM memory type register (CPUROM_MEMTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: system memory

1: system memory present on this bus

CPU ROM CoreSight peripheral identity register 4 (CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: ARM® JEDEC continuation code

CPU ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00C9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0xC9: Cortex®-M33

CPU ROM CoreSight peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: ARM® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x4: Cortex®-M33

CPU ROM CoreSight peripheral identity register 2 (CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CPU ROM CoreSight peripheral identity register 3 (CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CPU ROM CoreSight component identity register 0 (CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

CPU ROM CoreSight peripheral identity register 1 (CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

CPU ROM CoreSight component identity register 2 (CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

CPU ROM CoreSight component identity register 3 (CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

59.5.6 Processor ROM table register map

Table 730. CPU ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	CPUROM_ MEMTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD4 to FDC	Reserved																																
0xFD0	CPUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	CPUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	0	1	0	0	1
0xFE4	CPUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	CPUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CPUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CPUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CPUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	CPUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CPUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 727* for register boundary addresses.

59.6 Data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as one of the following:

- Watchpoint
- ETM trigger
- PC sampling trigger
- Data address sampling trigger
- Data comparator (COMP 1 only)
- Clock cycle counter comparator (COMP 0 only)

It also contains counters for:

- Clock cycles
- Folded instructions
- Load store unit (LSU) operations
- Sleep cycles
- Number of cycles per instruction
- Interrupt overhead

A DWT comparator compares the value held in its *DWT comparator x register (DWT_COMPxR)* with one of the following:

- A data address
- An instruction address
- A data value
- The cycle-count value, for COMP 0 only

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- One or more DWT data trace packets, containing one or more of:
  - The address of the instruction that caused a data access
  - An address offset, bits[15:0] of the data access address
  - The matched data value
- A watchpoint debug event, on either the PC value or the accessed data address
- A CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter the debug state.

For more details on how to use the DWT, refer to the Arm®v8-M Architecture Reference Manual *[4]*.

59.6.1 DWT registers

The DWT registers are located at the address range 0xE000 1000 to 0xE000 1FFC.

DWT control register (DWT_CTRLR)

Address offset: 0x000

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: number of comparators implemented (read only)
0x4: four comparators

Bit 27 **NOTRCPKT**: trace sampling and exception tracing support (read only)
0: supported

Bit 26 **NOEXTTRIG**: external match signal, CMPMATCH support (read only)
0: supported

Bit 25 **NOCYCCNT**: cycle counter support (read only)
0: supported

Bit 24 **NOPRFCNT**: profiling counter support (read only)
0: supported

Bit 23 **CYCDISS**: cycle counter disabled secure.
Controls whether the cycle counter is disabled in secure mode.
0: no effect
1: disable incrementing of the cycle counter when the processor is in secure state

Bit 22 **CYCEVTENA**: enable for POSTCNT underflow event counter packet generation
0: disabled
1: enabled

Bit 21 **FOLDEVTENA**: enable for folded instruction counter overflow event generation
0: disabled
1: enabled

Bit 20 **LSUEVTENA**: enable for LSU counter overflow event generation
0: disabled
1: enabled

Bit 19 **SLEEPEVTENA**: enable for sleep counter overflow event generation
0: disabled
1: enabled

Bit 18 **EXCEVTENA**: enable for exception overhead counter overflow event generation
0: disabled
1: enabled

Bit 17 **CPIEVTENA**: enable for CPI counter overflow event generation
0: disabled
1: enabled

Bit 16 **EXCTRCENA**: enable for exception trace generation
0: disabled
1: enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: enable for POSTCNT counter to be used as a timer for periodic PC sample packet generation
0: disabled
1: enabled

Bits 11:10 **SYNCTAP[1:0]**: position of the synchronization packet counter tap on the CYCCNT counter
This field determines the synchronization packet rate.
00: disabled, no synchronization packets
01: Tap at CYCCNT[24]
10: Tap at CYCCNT[26]
11: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Selects the position of the POSTCNT tap on the CYCCNT counter.
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: initial value of the POSTCNT counter
Writes to this field are ignored if POSTCNT counter is enabled. CYCEVTENA or PCSAMPLENA bits must be reset prior to writing POSTINIT.

Bits 4:1 **POSTRESET[3:0]**: reload value of the POSTCNT counter

Bit 0 **CYCCNTENA**: enable CYCCNT counter
0: disabled
1: enabled

DWT cycle count register (DWT_CYCCNTR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CYCCNT[31:0]**: processor clock-cycle counter

DWT CPI count register (DWT_CPICNTR)

Address offset: 0x008

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter

Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNTR, and counts any instruction fetch stalls.

DWT exception count register (DWT_EXCCNTR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: exception overhead cycle counter

Counts the number of cycles spent in exception processing.

DWT sleep count register (DWT_SLPCNTR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (DWT_LSUCNTR)

Address offset: 0x014

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: load store counter
Counts additional cycles required to execute load and store instructions.

DWT fold count register (DWT_FOLDCNTR)

Address offset: 0x018

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: folded instruction counter
Increments on each instruction that takes 0 cycles.

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EIASAMPLE[31:0]**: executed instruction address sample value.
Samples the current value of the program counter.

DWT comparator x register (DWT_COMPxR)

Address offset: 0x020 + 0x010 * x, (x = 0 to 3)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **COMP[31:0]**: reference value for comparison

DWT function register 0 (DWT_FUNCTR0)

Address offset: 0x028

Reset value: 0x5800 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 0.
0b01011: Cycle Counter, Instruction Address, Data Address and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 0.
For possible values of this field, refer to *[4]*.

DWT function register 1 (DWT_FUNCTR1)

Address offset: 0x038

Reset value: 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for match for comparator 1.
0b11010: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 1.
For possible values of this field, refer to *[4]*.

DWT function register 2 (DWT_FUNCTR2)

Address offset: 0x048

Reset value: 0x5000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification
Identifies the capability for MATCH for comparator 2
0b01010: Instruction Address, Data Address, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match

Indicates if a comparator match has occurred since the register was last read.

0: no match

1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Data value size:

Defines the size of the object being watched for by Data Value and Data Address comparators.

0x0: 1 byte

0x1: 2 bytes

0x2: 4 bytes

0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match

0x0: trigger only

0x1: Generate debug event

0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.

0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type

Controls the type of match generated by comparator 2.

For possible values of this field, refer to *[4]*

DWT function register 3 (DWT_FUNCTR3)

Address offset: 0x058

Reset value: 0xF000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: capability identification

Identifies the capability for MATCH for comparator 2.

0b11110: Instruction Address, Instruction Address Limit, Data Address, Data Address Limit, Data value, Linked Data Value, and Data Address With Value

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: comparator match
Indicates if a comparator match has occurred since the register was last read.
0: no match
1: a match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: data value size
Defines the size of the object being watched for by Data Value and Data Address comparators.
0x0: 1 byte
0x1: 2 bytes
0x2: 4 bytes
0x3: reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: action on match
0x0: trigger only
0x1: Generate debug event
0x2: For a Cycle Counter, Instruction Address, Data Address, Data Value or Linked Data Value comparator, generate a Data Trace Match packet. For a Data Address With Value comparator, generate a Data Trace Data Value packet.
0x3: For a Data Address Limit comparator, generate a Data Trace Data Address packet. For a Cycle Counter, Instruction Address Limit, or Data Address comparator, generate a Data Trace PC Value packet. For a Data Address With Value comparator, generate both a Data Trace PC Value packet and a Data Trace Data Value packet.

Bits 3:0 **MATCH[3:0]**: match type
Controls the type of match generated by comparator 2.
For possible values of this field, refer to *[4]*

DWT device type architecture register (DWT_DEVARCHR)

Address offset: 0xFC8

Reset value: 0x4770 1A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DWT_DEVARCH register present
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: DWT architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: DWT architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA02: DWT architecture

DWT device type register (DWT_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-type
0x0: other

Bits 3:0 **MAJOR[3:0]**: major type
0x0: miscellaneous

DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: Cortex®-M33 DWT part number

DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: Cortex®-M33 DWT part number

DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: No customer modifications

DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: Common identification value

DWT CoreSight peripheral identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

59.6.2 DWT register map

The DWT registers are located at the address range 0xE000 1000 to 0xE000 1FFC.

Table 731. DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DWT_CTRLR	NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTPRESET[3:0]				CYCCNTENA
	Reset value	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DWT_CYCCNTR	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x00C	DWT_EXCCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x010	DWT_SLPCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	DWT_LSUCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x018	DWT_FOLDCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									X	X	X	X	X	X	X	X

Table 731. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x020	DWT_COMP0R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x024	Reserved																																
0x028	DWT_FUNCTR0	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	1			0													0	0					0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	DWT_COMP1R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x034	Reserved																																
0x038	DWT_FUNCTR1	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	1	1	0	1	0			0													0	0					0	0	0	0	0	0
0x03C	Reserved																																
0x040	DWT_COMP2R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x044	Reserved																																
0x048	DWT_FUNCTR2	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x04C	Reserved																																
0x050	DWT_COMP3R	COMP[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x054	Reserved																																
0x058	DWT_FUNCTR3	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
	Reset value	1	1	1	1	0			0													0	0					0	0	0	0	0	0
0x05C to 0xFC4	Reserved																																
0xFC8	DWT_DEVARCHR	ARCHITECT[10:0]											PRESENT	REVISION[3:0]				ARCHVER[3:0]				ARCHPART[11:0]												
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0

Table 731. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	DWT_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	DWT_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	DWT_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 727: Processor ROM table* for register boundary addresses.

59.7 Instrumentation trace macrocell (ITM)

The ITM generates trace information in packets. Three sources can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The three sources in decreasing order of priority are the following:

- Software trace

  The software can write directly to any of the 32 x 32-bit ITM stimuli registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access and the data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.
- Hardware trace

  The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.
- Local timestamping

  The ITM contains a 21-bit counter clocked by the (predivided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

For more information on the ITM and how to use it, refer to *[4]*.

59.7.1 ITM registers

The ITM registers are located at the address range 0xE000 0000 to 0xE000 0FFC.

ITM stimulus register x (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when writing

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	rw	rw

Bits 31:0 **STIMULUS[31:0]**: trace output data
write data is output on the trace bus as a software event packet.

ITM stimulus register x [alternate] (ITM_STIMRx)

Address offset: 0x000 + 0x004 * x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

Condition: when reading

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DISABLE	FIFO_READY
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DISABLE**: Disable flag
0: stimulus port and ITM enabled
1: stimulus port and ITM disabled

Bit 0 **FIFO_READY**: FIFO ready indicator
0: stimulus port buffer is full (or port is disabled)
1: stimulus port can accept new write data

ITM trace enable register (ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: stimulus port enable
Each bit x(0 to 31) enables the stimulus port associated with the ITM_STIMRx register.
0: port disabled
1: port enabled

ITM trace privilege register (ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: disable unprivileged access to ITM stimulus ports
Each bit controls eight stimulus ports.
XXX0: unprivileged access permitted on ports 0 to 7
XXX1: only privileged access permitted on ports 0 to 7
XX0X: unprivileged access permitted on ports 8 to 15
XX1X: only privileged access permitted on ports 8 to 15
X0XX: unprivileged access permitted on ports 16 to 23
X1XX: only privileged access permitted on ports 16 to 23
0XXX: unprivileged access permitted on ports 24 to 31
1XXX: only privileged access permitted on ports 24 to 31

ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE[1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
						rw	rw			rw	r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: indicates whether the ITM is currently processing events
0: not busy
1: busy

Bits 22:16 **TRACEBUSID[6:0]**: identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field.
Note: Different identifiers must be used for each trace source in the system.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **TSPRESCALE[1:0]**: local timestamp prescaler, used with the trace packet reference clock
0x0: no prescaling
0x1: Divide by 4.
0x2: Divide by 16.
0x3: Divide by 64.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STALLENA**: stall enable
0: Drop hardware source packets and generate an overflow if the ITM output is stalled.
1: Stall the processor to guarantee delivery of data trace packets.

Bit 4 **SWOENA**: SWO enable
Enables asynchronous clocking of the timestamp counter (read only).
0: Timestamp counter uses processor clock.

Bit 3 **TXENA**: transmit enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: disabled
1: enabled

Bit 2 **SYNCENA**: synchronization packet transmission enable
The debugger setting this bit must also configure the DWT_CTRLR.SYNCTAP field for the correct synchronization speed.
0: disabled
1: enabled

Bit 1 **TSENA**: local timestamp generation enable
0: disabled
1: enabled

Bit 0 **ITMENA**: ITM enable
0: disabled
1: enabled

ITM device type architecture register (ITM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: ITM architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA01: ITM architecture

ITM device type register (ITM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-type
0x4: associated with a bus, stimulus derived from bus activity

Bits 3:0 **MAJOR[3:0]**: major type
0x3: trace source

ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: ITM part number

ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: ITM part number

ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component identification bits [7:0]
0x0D: Common identification value

ITM CoreSight peripheral identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component identification bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component identification bits [11:8]
0x0: Common identification value

ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component identification bits [23:16]
0x05: Common identification value

ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component identification bits [31:24]
0xB1: Common identification value

59.7.2 ITM register map

The ITM registers are located at the address range 0xE000 0000 to 0xE000 0FFC.

Table 732. ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 to 0x07C	ITM_STIMR0 to ITM_STIMR31	STIMULUS[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x07C to 0xDFC	Reserved																																
0xE00	ITM_TER	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04 to 0xE3C	Reserved																																
0xE40	ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK [3:0]			
	Reset value																													0	0	0	0
0xE44 to 0xE7C	Reserved																																
0xE80	ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	TSPRESCALE [1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0							0	0			0	0	0	0	0	0
0xE84 to 0xFB8	Reserved																																

Table 732. ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFBC	ITM_DEVARCHR	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[3:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0xFC0 to 0xFC8	Reserved																																
0xFCC	ITM_DEVTYPER	Res.																								SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	0	0	0	0	1	1
0xFD0	ITM_PIDR4	Res.																								SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	ITM_PIDR0	Res.																								PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ITM_PIDR1	Res.																								JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ITM_PIDR2	Res.																								REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	ITM_PIDR3	Res.																								REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ITM_CIDR0	Res.																								PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ITM_CIDR1	Res.																								CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.																								PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.																								PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 727: Processor ROM table* for register boundary addresses.

59.8 Breakpoint unit (BPU)

The BPU allows the user to set hardware breakpoints. It contains eight comparators that monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

For more information on the breakpoint unit and how to use it, refer to *[4]*.

59.8.1 BPU registers

The BPU registers are located at the address range 0xE0002000 to 0xE0002FFC.

BPU control register (BPU_CTRLR)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			Res.	Res.	Res.	Res.	NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r					r	r	r	r			rw	rw

Bits 31:28 **REV[3:0]**: revision number
0x1: BPU version 2

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:12, 7:4 **NUM_CODE[6:0]**: number of instruction address comparators supported
0x08: 8 instruction comparators supported

Bits 11:8, 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key
A write to FPB_CTRLR register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: FPB enable
0: disabled
1: enabled

BPU comparator x register (BPU_COMPxR)

Address offset: 0x008 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPADDR[15:1]															BE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]**: breakpoint address

Bit 0 **BE**: breakpoint enable
0: disabled
1: enabled

BPU device type architecture register (BPU_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4770 1A03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DEVARCH register present
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x0: BPU architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x1: BPU architecture v2.0

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA03: BPU architecture

BPU device type register (BPU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: sub-type
0x0: other

Bits 3:0 **MAJOR[3:0]**: major type
0x0: miscellaneous

BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: BPU part number

BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: BPU part number

BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm ® JEDEC code

BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

BPU CoreSight peripheral identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: debug component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

BPU CoreSight component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

BPU CoreSight component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

59.8.2 BPU register map

The BPU registers are located at the address range 0xE000 2000 to 0xE000 2FFC.

Table 733. BPU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	BPU_CTRLR	REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE [6:4]			Res.	Res.	Res.	Res.	NUM_CODE [3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0					1	0	0	0			0	0
0x004	Reserved																																
0x008 to 0x024	BPU_COMP0R to BPU_COMP7R	BPADDR[31:1]																															BE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028 to 0xFB8	Reserved																																
0xFBC	BPU_DEVARCHR	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	1
0xFC0 to 0xFC8	Reserved																																
0xFCC	BPU_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	BPU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	BPU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	BPU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	BPU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	BPU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	BPU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	BPU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0

Table 733. BPU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	BPU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	BPU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 727: Processor ROM table* for register boundary addresses.

59.9 Embedded Trace Macrocell (ETM)

The ETM is a CoreSight™ component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex®-M33 core to be traced. In the STM32H5 devices, the ETM is configured for instruction trace only. Data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- Number of instructions executed in the same cycle
- Changes in program flow
- Current processor instruction state
- Addresses of memory locations accessed by load and store instructions
- Type, direction, and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM-M33 Technical Reference Manual *[6]*.

59.9.1 ETM registers

The ETM registers are located at the address range 0xE004 1000 to 0xE004 1FFC.

ETM programming control register (ETM_PRGCTLR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: trace unit enable
0: disabled
1: enabled

ETM status register (ETM_STATR)

Address offset: 0x00C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: stability status
Indicates that the ETM-M33 registers are stable and can be read.
0: not stable
1: stable

Bit 0 **IDLE**: trace unit status
Indicates that the trace unit is inactive.
0: not idle
1: idle

ETM configuration register (ETM_CONFIGR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
			rw		rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RS**: return stack enable
0: disabled
1: enabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:5 **COND[5:0]**: conditional instruction tracing
0x0: conditional instruction tracing disabled
0x1: conditional load instructions traced
0x2: conditional store instructions traced
0x3: conditional load and store instructions traced
0x7: All conditional instructions traced

Bit 4 **CCI**: cycle counting in instruction trace
0: disabled
1: enabled

Bit 3 **BB**: branch broadcast mode
0: disabled
1: enabled

Bits 2:0 Reserved, must be kept at reset value.

ETM event control 0 register (ETM_EVENTCTL0R)

Address offset: 0x020

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
rw				rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: resource type for event1
0: single selected resource
1: boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: resource number based on TYPE1
Selects the resource number, based on the value of TYPE1.
When TYPE1 = 0, a single resource from 0-15 defined by SEL1[3:0] is selected.
When TYPE1 = 1, a boolean combined resource pair defined by SEL1[2:0] is selected.

Bit 7 **TYPE0**: resource type for event0
0: single selected resource
1: boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: resource number based on TYPE0
Selects the resource number, based on the value of TYPE0.
When TYPE0 = 0, a single resource from 0-15 defined by SEL0[3:0] is selected.
When TYPE0 = 1, a boolean combined resource pair defined by SEL0[2:0] is selected.

ETM event control 1 register (ETM_EVENTCTL1R)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
			rw	rw										rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: low-power state behavior override
0: normal low-power state behavior
1: The resources and event trace generation are not affected by entry to a low-power state.

Bit 11 **ATB**: ATB trigger enable
0: disabled
1: enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: instruction event generation
Enables generation of an event element in the instruction stream.
0bX0: Event0 does not cause an event element.
0bX1: Event0 causes an event element when it occurs.
0b0X: Event1 does not cause an event element.
0b1X: Event1 causes an event element when it occurs.

ETM stall control register (ETM_STALLCTLR)

Address offset: 0x02C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
					rw		rw					rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **INSTPRIORITY**: instruction trace priority
Prioritizes instruction trace if instruction trace buffer space is less than LEVEL[3:0].
0: The ETM must not prioritize instruction trace.
1: The ETM can prioritize instruction trace.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ISTALL**: processor stalling
Stalls processor based on instruction trace buffer space.
0: The ETM must not stall the processor.
1: The ETM can stall the processor.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **LEVEL[3:0]**: Threshold at which stalling becomes active
This field provides four levels. This level can be varied to optimize the level of invasion caused by stalling, balanced against the risk of a FIFO overflow.
0x0: zero invasion, but greater risk of FIFO overflow
...
0xF: maximum invasion but less risk of FIFO overflow

ETM synchronization period register (ETM_SYNCPR)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: synchronization period
Defines the number of bytes of trace between trace synchronization requests as a total of the number of bytes generated by the instruction stream.
0xA: 1024 bytes

ETM cycle count control register (ETM_CCCTLR)

Address offset: 0x038

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: instruction trace cycle count threshold

Sets the threshold value for instruction trace cycle counting. The threshold represents the minimum interval between cycle-count trace packets.

ETM trace identification register (ETM_TRACEIDR)

Address offset: 0x040

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace identification to output onto the trace bus

This field must be programmed with a unique value to differentiate it from other trace sources in the system.

Values 0x00 and 0x70-0x7F are reserved.

ETM ViewInst main control register (ETM_VICTLR)

Address offset: 0x080

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: exception level in secure state
Controls whether instruction tracing is enabled for the corresponding exception level, in secure state.
0bXXX0: instruction trace not generated in secure state, for exception level 0
0bXXX1: instruction trace generated in secure state, for exception level 0
0b0XXX: instruction trace not generated in secure state, for exception level 3
0b1XXX: instruction trace generated in secure state, for exception level 3

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: trace system error exception
0: The system error exception is traced only if the instruction or exception immediately before the system error exception is traced.
1: The system error exception is always traced.

Bit 10 **TRCRESET**: trace reset exception
0: The reset exception is traced only if the instruction or exception immediately before the reset exception is traced.
1: The reset exception is always traced.

Bit 9 **SSSTATUS**: start/stop logic status
0: stopped
1: started

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: event selector

ETM counter reload value register 0 (ETM_CNTRLDVR0)

Address offset: 0x140

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: counter reload value
This value is loaded in to the counter each time the reload event occurs.

ETM identification register 8 (ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXSPEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXSPEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MAXSPEC[31:0]**: maximum speculation depth

Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.

0x0: The maximum trace speculation depth is zero.

ETM identification register 9 (ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: number of P0 right-hand keys used

0x0: no P0 right-hand keys used in instruction trace

ETM identification register 10 (ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: number of P1 right-hand keys used (including normal and special keys)

0x0: no P1 right-hand keys used in instruction trace

ETM identification register 11 (ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: number of special P1 right-hand keys used
0x0: no special P1 right-hand keys used in any configuration

ETM identification register 12 (ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDKEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDKEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDKEY[31:0]**: number of conditional instruction right-hand keys used (including normal and special keys)
0x1: one conditional instruction right-hand key implemented

ETM identification register 13 (ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDSPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDSPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDSPC[31:0]**: number of special conditional instruction right-hand keys used
0x0: no special conditional instruction right-hand keys implemented

ETM implementation specific register 0 (ETM_IMSPECR0)

Address offset: 0x1C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SUPPORT[3:0]**: implementation specific extension support
0x0: no implementation specific extensions are supported

ETM identification register 0 (ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x2800 06E1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1]
		r												r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
r		r	r	r	r	r		r	r	r	r	r	r	r	

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: commit field meaning
Indicates the meaning of the commit field in some packets.
1: commit mode 1

Bits 28:18 Reserved, must be kept at reset value.

Bit 17 **TRCEXDATA**: trace data transfers for exceptions
Indicates support for the tracing of data transfers for exceptions and exception returns.
0: not implemented

Bits 16:15 **QSUPP[1:0]**: Q element support
0: not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: conditional results tracing
Indicates how conditional results are traced.
0: The trace unit indicates only if a conditional instruction passes or fails its condition code check

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported
0x1: two events

Bit 9 **RETSTACK**: return stack support
1: two entry return stacks

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: cycle counting support
1: cycle counting implemented

Bit 6 **TRCCOND**: conditional instruction support
1: conditional instruction tracing implemented

Bit 5 **TRCBB**: branch broadcast support
1: branch broadcast tracing implemented

Bits 4:3 **TRCDATA[1:0]**: data tracing support
0x0: data tracing not supported

Bits 2:1 **INSTP0[1:0]**: support for tracing of load and store instructions as P0 elements
0x0: not supported

Bit 0 Reserved, must be kept at reset value.

ETM identification register 1 (ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F421

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: trace unit designer
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: major trace unit architecture version number
0x4: ETMv4

Bits 7:4 **TRCARCHMIN[3:0]**: minor trace unit architecture version number
0x2: minor revision 2

Bits 3:0 **REVISION[3:0]**: implementation revision number
0x1: implementation revision 1

ETM identification register 2 (ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: cycle counter size
0x0: 12 bits

Bits 24:20 **DVSIZE[4:0]**: data value size
0x0: data value size not supported

Bits 19:15 **DASIZE[4:0]**: data address size.
0x0: data address size not supported

Bits 14:10 **VMIDSIZE[4:0]**: virtual machine ID size
0x0: virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: context ID size
0x0: context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: instruction address size
0x4: maximum 32-bit address size

ETM identification register 3 (ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0F09 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: ETM_STALLCTLR.NOOVERFLOW implementation
0: not implemented

Bits 30:28 **NUMPROC[2:0]**: number of processors available for tracing
0x0: one processor

Bit 27 **SYSSTALL**: system support for stall control of the processor
1: system supports stall control

Bit 26 **STALLCTL**: stall control support
1: ETM_STALLCTLR implemented

Bit 25 **SYNCPR**: trace synchronization period support
1: ETM_SYNCPR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24 **TRCERR**: ETM_VICTLR.TRCERR implementation
0x1: implemented

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: privilege levels implementation
0x9: privilege levels thread and handler implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: minimum value that can be programmed to TRCCCCTLR.THRESHOLD
Defines the minimum cycle counting threshold.
0x4: minimum of four-instruction trace cycles

ETM identification register 4 (ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0011 4000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: number of virtual machine ID (VMID) comparators
0x0: VMID comparators not implemented

Bits 27:24 **NUMCIDC[3:0]**: number of context ID comparators
0x0: context ID comparators not supported

Bits 23:20 **NUMSSCC[3:0]**: number of single-shot comparator controls
0x1: one single-shot comparator control implemented

Bits 19:16 **NUMRSPAIR[3:0]**: number of resource selection pairs
0x1: two resource selection pairs implemented

Bits 15:12 **NUMPC[3:0]**: number of processor comparator inputs for the DWT
0x4: four processor comparator inputs implemented

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPDAC**: data address comparisons
0: data address comparisons not supported

Bits 7:4 **NUMDVC[3:0]**: number of data value comparators
0x0: no data value comparators implemented

Bits 3:0 **NUMACPAIRS[3:0]**: number of address comparator pairs
0x0: no address comparator pairs implemented

ETM identification register 5 (ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: reduced function counter
1: counter 0 implemented as a reduced function counter

Bits 30:28 **NUMCNTR[2:0]**: number of counters
0x1: one counter implemented.

Bits 27:25 **NUMSEQSTATE[2:0]**: number of sequencer states
0x0: no sequencer states implemented.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: low-power state override support
1: low-power state override support implemented

Bit 22 **ATBTRIG**: ATB trigger support
1: ATB trigger support implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: number of bits of trace identification
0x7: 7-bit trace identification implemented

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: number of external input selectors
0x0: no external input selectors implemented.

Bits 8:0 **NUMEXTIN[8:0]**: number of external inputs
0x004: four external inputs implemented.

ETM resource register 2 (ETM_RSCTLR2)

Address offset: 0x208

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: result of a combined pair of resources inversion
0: not inverted
1: inverted

Bit 20 **INV**: selected resources inversion
0: not inverted
1: inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: group of resources selection
0x0: external input selectors (select 0-3)
0x1: inputs from processor DWT comparators element (select 0-3)
0x2: counter at zero (select 0)
0x3: single-shot comparator (select 0)
Others: reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: more resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM resource register 3 (ETM_RSCTLR3)

Address offset: 0x20C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: selected resources inversion
0: not inverted
1: inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: group of resources selection
0x0: external input selectors (select 0-3)
0x1: inputs from processor DWT comparators element (select 0-3)
0x2: counter at zero (select 0)
0x3: single-shot comparator (select 0)
Others: reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: more resources selection
Selects one or more resources from the group selected in GROUP[2:0].

ETM single-shot comparator control register 0 (ETM_SSCCR0)

Address offset: 0x280

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: single-shot comparator reset
Enables the single-shot comparator resource to be reset when it occurs, to enable another comparator match to be detected.
1: reset enabled

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (ETM_SSCSR0)

Address offset: 0x2A0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
												r	r	r	r

Bit 31 **STATUS**: single-shot comparator status
Indicates whether any of the selected comparators have matched.
0: no match occurred
1: at least one match occurred

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **PC**: processor comparator input sensitivity
1: single-shot comparator sensitive to processor comparator inputs

Bit 2 **DV**: data value comparator support
0: single-shot data value comparisons not supported

Bit 1 **DA**: data address comparator support
0: single-shot data address comparisons not supported

Bit 0 **INST**: instruction address comparator support
0: single-shot instruction address comparisons not supported

ETM single-shot processor comparator input control register 0 (ETM_SSPCICR0)

Address offset: 0x2C0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PC[3:0]**: processor comparator inputs selection for single-shot control
0xXXX0: processor comparator input 0 not selected
0xXXX1: processor comparator input 0 selected
0xXX0X: Processor comparator input 1 not selected
0xXX1X: processor comparator input 1 selected
0xX0XX: processor comparator input 2 not selected
0xX1XX: processor comparator input 2 selected
0x0XXX: processor comparator input 3 not selected
0x1XXX: processor comparator input 3 selected

ETM power-down control register (ETM_PDCR)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												rw			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: power-up request
0: power-up not requested
1: power-up requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (ETM_PDSR)

Address offset: 0x314

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: sticky power-down state
0: Trace register power has not been removed since the ETM_PDSR was last read.
1: Trace register power has been removed since the ETM_PDSR was last read.

Bit 0 **POWER**: ETM power-up status
1: ETM powered up

ETM claim tag set register (ETM_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: claim tag bits setting

Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: claim tag bits reset

Write:
0000: no effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

ETM authentication status register (ETM_AUTHSTATR)

Address offset: 0xFB8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: security level for secure non-invasive debug
0x2: secure non-invasive debug disabled
0x3: secure non-invasive debug enabled

Bits 5:4 **SID[1:0]**: security level for secure invasive debug
0x0: not implemented

Bits 3:2 **NSNID[1:0]**: security level for non-secure non-invasive debug
0x2: non-secure non-invasive debug disabled
0x3: non-secure non-invasive debug enabled

Bits 1:0 **NSID[1:0]**: security level for non-secure invasive debug
0x0: not implemented

ETM device type architecture register (ETM_DEVARCHR)

Address offset: 0xFBC

Reset value: 0x4772 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: architect JEP106 code
0x23B: JEP106 continuation code 0x4, JEP106 ID code 0x3B. Arm® limited.

Bit 20 **PRESENT**: DEVARCH register presence
0x1: present

Bits 19:16 **REVISION[3:0]**: architecture revision
0x2: ETM architecture v4.2

Bits 15:12 **ARCHVER[3:0]**: architecture version
0x4: ETM architecture v4.2

Bits 11:0 **ARCHPART[11:0]**: architecture part
0xA13: ETM architecture

ETM CoreSight device type register (ETM_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: device sub-type identifier
0x1: processor trace

Bits 3:0 **MAJORTYPE[3:0]**: device main type identifier
0x3: trace source

ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: ETM part number

ETM CoreSight peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: ETM part number

ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

ETM CoreSight peripheral identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: trace generator component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

ETM CoreSight component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

59.9.2 ETM register map

The ETM registers are accessed by the debugger at the address range 0xE0041000 to 0xE0041FFC.

Table 734. ETM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETM_PRGCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	Reserved																																
0x00C	ETM_STATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value																															X	X
0x010	ETM_CONFIGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RS	Res.	COND[5:0]						CCI	BB	Res.	Res.	Res.
	Reset value																				X		X	X	X	X	X	X	X	X			
0x014 to 0x01C	Reserved																																
0x020	ETM_ EVENTCTL0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE1	Res.	Res.	Res.	SEL1[3:0]				TYPE0	Res.	Res.	Res.	SEL0[3:0]			
	Reset value																	X				X	X	X	X	X				X	X	X	X
0x024	ETM_ EVENTCTL1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
	Reset value																				X	X										X	X
0x028	Reserved																																

Table 734. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	ETM_STALLCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTPRIORITY	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[3:0]			
	Reset value																						X		X					X	X	X	X
0x030	Reserved																																
0x034	ETM_SYNCPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]			
	Reset value																												0	1	0	1	0
0x038	ETM_CCCTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					X	X	X	X	X	X	X	X	X	X	X	X
0x03C	Reserved																																
0x040	ETM_TRACEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										X	X	X	X	X	X	X
0x044 to 0x07C	Reserved																																
0x080	ETM_VICTLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
	Reset value													X	X	X	X					X	X	X		X	X	X	X	X	X	X	X
0x084 to 0x13C	Reserved																																
0x140	ETM_CNTRLDVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x144 to 0x17C	Reserved																																
0x180	ETM_IDR8	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	ETM_IDR9	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	ETM_IDR10	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	ETM_IDR11	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	ETM_IDR13	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198 to 0x1BC	Reserved																																
0x1C0	ETM_IMSPECR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUPPORT [3:0]			
	Reset value																													0	0	0	0
0x1C4 to 0x1DC	Reserved																																

Table 734. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1E0	ETM_IDR0	Res.	Res.	COMMOPT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCEXDATA	QSUPP[1:0]		Res.	CONDTYPE [1:0]		NUMEVENT [1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
	Reset value			1												0	0	0		0	0	0	1	1		1	1	1	0	0	0	0	
0x1E4	ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ [3:0]				TRCARCHMIN [3:0]			REVISION [3:0]				
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	0	1	0	0	0	0	1
0x1E8	ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]					VMIDSIZE[4:0]			CIDSIZE[4:0]				IASIZE[4:0]							
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	Res.	Res.	CCITMIN[11:0]											
	Reset value	0	0	0	0	1	1	1	1					1	0	0	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F0	ETM_IDR4	NUMVMIDC [3:0]				NUMCIDC [3:0]				NUMSSCC [3:0]				NUMRSPAIR [3:0]				NUMPC [3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC [3:0]				NUMACPAIRS [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0				0	0	0	0	0	0	0	0	0
0x1F4	ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE [2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE [5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL [2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					0	0	0	0	0	0	0	0	0	1	0	0
0x1F8 to 0x204	Reserved																																
0x208	ETM_RSCTLR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											X	X		X	X	X									X	X	X	X	X	X	X	X
0x20C	ETM_RSCTLR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												X		X	X	X									X	X	X	X	X	X	X	X
0x210 to 0x27C	Reserved																																
0x280	ETM_SSCCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								X																								
0x284 to 0x29C	Reserved																																
0x2A0	ETM_SSCSR0	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
	Reset value	X																												X	X	X	X
0x2A4 to 0x2BC	Reserved																																

Table 734. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C0	ETM_SSPCICR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[3:0]			
	Reset value																													X	X	X	X
0x2C4 to 0x30C	Reserved																																
0x310	ETM_PDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
	Reset value																													0			
0x314	ETM_PDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318 to 0xF9C	Reserved																																
0xFA0	ETM_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFB4	Reserved																																
0xFB8	ETM_AUTHSTATR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									X	X	X	X	X	X	X	X
0xFBC	ETM_DEVARCHR	ARCHITECT[10:0]											PRESEN	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0to 0xFC8	Reserved																																
0xFCC	ETM_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYP [3:0]			
	Reset value																									0	0	0	1	0	0	1	1
0xFD0	ETM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4to 0xFDC	Reserved																																
0xFE0	ETM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	ETM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ETM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0

Table 734. ETM register map and reset values (continued)

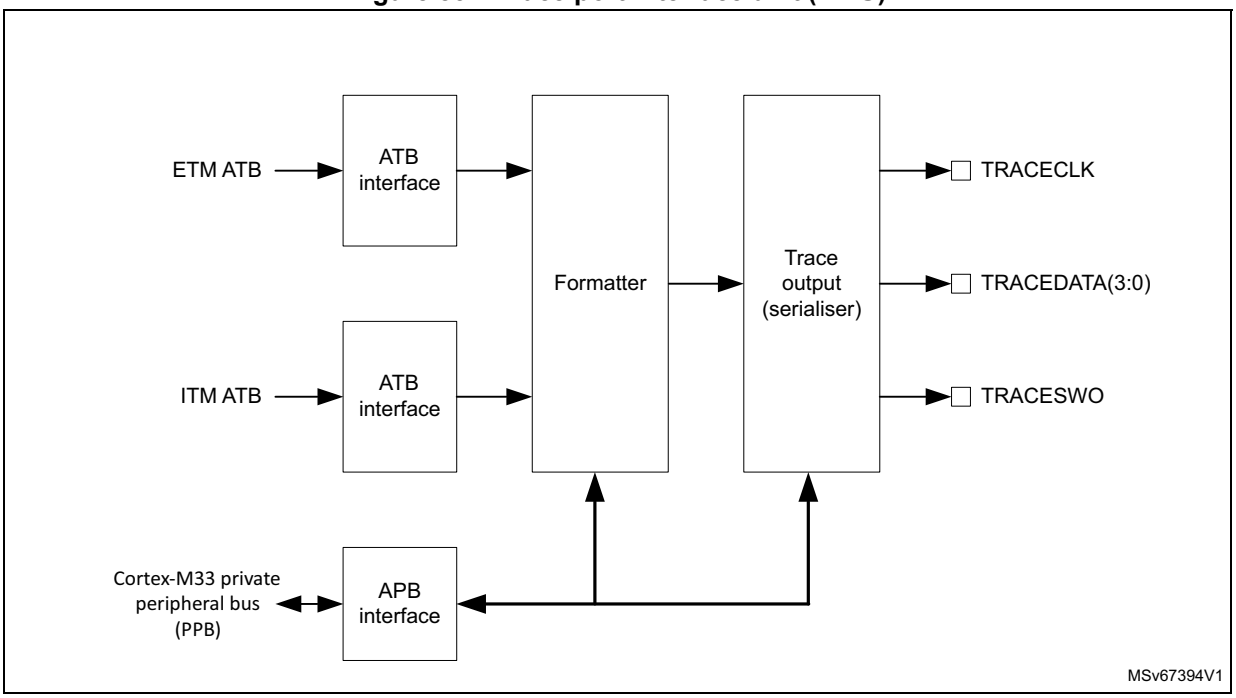
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF0	ETM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 727: Processor ROM table* for register boundary addresses.

59.10 Trace port interface unit (TPIU)

The TPIU formats the trace stream and outputs it on the external trace port signals. As shown in the figure below, the TPIU has two ATB slave ports for incoming trace data from the ETM and ITM respectively. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and four data outputs, TRACEDATA(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up I/Os for other purposes, at the expenses of bandwidth restriction of the trace port, and hence of the quantity of trace information that can be output in real time.

Figure 854. Trace port interface unit (TPIU)



Trace data can also be output on the serial-wire output, TRACESWO.

For more information on the trace port interface in the Cortex®-M33, refer to the Arm® Cortex®-M33 Technical Reference Manual *[5]*.

59.10.1 TPIU registers

TPIU supported port size register (TPIU_SSPSR)

Address offset: 0x000

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: trace port sizes, from 1 to 32 pins
Bit n-1 when set, indicates that port size n is supported.
0x0000 000F: port sizes 1 to 4 supported

TPIU current port size register (TPIU_CSPSR)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: current trace port size
Bit n-1 when set, indicates that the current port size is n pins. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result.
This register must only be modified when the formatter is stopped.

TPIU asynchronous clock prescaler register (TPIU_ACPR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: baud rate for the asynchronous output, TRACESWO
The baud rate is given by the TRACELKIN frequency divided by (PRESCALER +1).

TPIU selected pin protocol register (TPIU_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **TXMODE[1:0]**: protocol used for trace output
0x0: parallel trace port mode
0x1: asynchronous SWO using Manchester encoding
0x2: asynchronous SWO using NRZ encoding
0x3: reserved

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: formatter stop

Indicates whether formatter can be stopped or not.

1: The formatter cannot be stopped.

Bit 2 **TCPRESENT**: TRACECTL output pin availability

Indicates whether the optional TRACECTL output pin is available for use.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: formatter stop

The formatter has received a stop request signal and all trace data and post-amble is sent. Any additional trace data on the ATB interface is ignored.

0: The formatter has not stopped.

Bit 0 **FLINPROG**: flush in progress

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.

0: no flush in progress

1: flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0102

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
							r		rw					rw	

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **TRIGIN**: trigger on trigger in

1: Indicates a trigger in the trace stream when the TRIGIN input is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: flush on manual

0: flush completed

1: Generates a flush.

Bits 5:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** continuous formatting enable

Setting this bit to zero in SWO mode bypasses the formatter and only ITM/DWT trace is output, ETM trace is discarded.

0: continuous formatting disabled

1: continuous formatting enabled

Bit 0 Reserved, must be kept at reset value.

TPIU periodic synchronization counter register (TPIU_PSCR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PSCOUNT[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PSCOUNT[12:0]**: formatter frames counter

Enables effective use of different sized TPAs without wasting large amounts of the storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word synchronization frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSETR)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: claim tag bits setting
Write:
0000: no effect
xxx1: Sets bit 0.
xx1x: Sets bit 1.
x1xx: Sets bit 2.
1xxx: Sets bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: claim tag bits reset
Write:
0000: no effect
xxx1: Clears bit 0.
xx1x: Clears bit 1.
x1xx: Clears bit 2.
1xxx: Clears bit 3.
Read: Returns current value of claim tag.

TPIU device configuration register (TPIU_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0000 0CA1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: Serial-wire output, NRZ support
0x1: supported

Bit 10 **SWOMAN**: Serial-wire output, Manchester encoded format, support
0x1: supported

Bit 9 **TCLKDATA**: trace clock plus data support
0x0: supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 bytes

Bit 5 **CLKRELAT**: ATB clock and TRACECLKIN relationship (synchronous or asynchronous)
0x1: asynchronous

Bits 4:0 **MAXNUM[4:0]**: number/type of ATB input port multiplexing
0x1: two input ports

TPIU device type identifier register (TPIU_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: trace port component

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x1: trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

TPIU CoreSight peripheral identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component ID bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

59.10.2 TPIU register map

Table 735. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0x004	TPIU_CSPSR	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008	Reserved																																
0x010	TPIU_ACPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 to 0x0EC	Reserved																																
0x0F0	TPIU_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXMODE [1:0]	
	Reset value																															0	1
0x0F4 to 0x2FC	Reserved																																
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGIN	Res.	FONMAN	Res.	Res.	Res.	Res.	ENFCONT	Res.
	Reset value																								1		0					1	

Table 735. TPIU register map and reset values (continued)

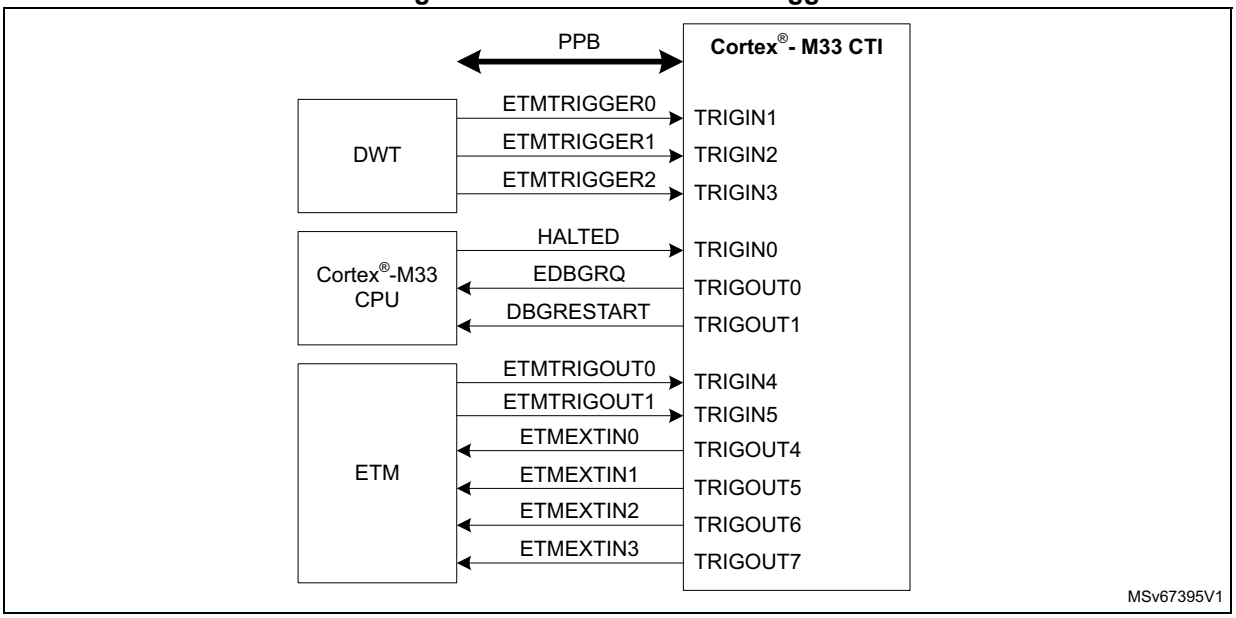
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x308	TPIU_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
030C to 0xF9C	Reserved																																
0xFA0	TPIU_CLAIMSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0FA8 to 0xFC4	Reserved																																
0xFC8	TPIU_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ:	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MaXNUM[4:0]				
	Reset value																					1	1	0	0	1	0	1	0	0	0	0	1
0xFCC	TPIU_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE0	TPIU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	0	1
0xFE4	TPIU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TPIU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 726: MCU ROM table* for register boundary addresses.

59.11 Cross-trigger interface (CTI)

The CTI (see *Figure 855*) allows cross triggering between the processor and the ETM.

Figure 855. Embedded cross trigger



The CTI enables events from various sources to trigger debug and/or trace activity. For example, a watchpoint reached in the processor can start or stop code trace, or a trace comparator can halt the processor.

The trigger input and output signals for the CTI are listed in the tables below.

Table 736. CTI inputs

Number	Source signal	Source component	Comments
0	HALTED	CPU	Processor halted - CPU is in debug mode
1	ETMTRIGGER0	DWT	DWT comparator output 0
2	ETMTRIGGER1	DWT	DWT comparator output 1
3	ETMTRIGGER2	DWT	DWT comparator output 2
4	ETMTRIGOUT0	ETM	ETM event output 0
5	ETMTRIGOUT1	ETM	ETM event output 1
6	-	-	Not used
7	-	-	Not used

Table 737. CTI outputs

Number	Source signal	Destination component	Comments
0	EDBGRQ	CPU	CPU halt request - Puts CPU in debug mode
1	DBGRESTART	CPU	CPU restart request - CPU exits debug mode

Table 737. CTI outputs (continued)

Number	Source signal	Destination component	Comments
2	ETMEXTIN0	ETM	ETM event input 0
3	ETMEXTIN1	ETM	ETM event input 1
4	ETMEXTIN2	ETM	ETM event input 2
5	ETMEXTIN3	ETM	ETM event input 3
6	-	-	Not used
7	-	-	Not used

For more information on the cross-trigger interface CoreSight™ component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual *[2]*.

59.11.1 CTI registers

The register file base address for the CTI is 0xE004 2000.

CTI control register (CTI_CONTROLR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: global CTI enable
0: disabled
1: enabled

CTI trigger acknowledge register (CTI_INTACKR)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: trigger acknowledge

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSETR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: channel event setting
Read:
XXX0: channel 0 event inactive
XXX0: channel 0 event active
XX0X: channel 1 event inactive
XX1X: channel 1 event active
X0XX: channel 2 event inactive
X1XX: channel 2 event active
0XXX: channel 3 event inactive
1XXX: channel 3 event active
Write:
XXX0: no effect
XXX0: Sets event on channel 0.
XX0X: no effect
XX1X: Sets event on channel 1.
X0XX: no effect
X1XX: Sets event on channel 2.
0XXX: no effect
1XXX: Sets event on channel 3.

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: channel event clear
0000: no effect
XXX1: Clears event on channel 0.
XX1X: Clears event on channel 1.
X1XX: Clears event on channel 2.
1XXX: Clears event on channel 3.

CTI application pulse register (CTI_APPPULSER)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: pulse channel event
This register clears itself immediately.
0000: no effect
XXX1: Generates pulse on channel 0.
XX1X: Generates pulse on channel 1.
X1XX: Generates pulse on channel 2.
1XXX: Generates pulse on channel 3.

CTI trigger input x enable register (CTI_INENxR)

Address offset: 0x020 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: trigger input event enable

Enables or disables a cross trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

0000: Trigger does not generate events on channels.

XXX1: Trigger x generates events on channel 0.

XX1X: Trigger x generates events on channel 1.

X1XX: Trigger x generates events on channel 2.

1XXX: Trigger x generates events on channel 3.

CTI trigger output x enable register (CTI_OUTENxR)

Address offset: 0x0A0 + 0x004 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: trigger output event enable

For each channel, defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

0000: Channel events do not generate triggers on trigger outputs.

XXX1: Channel 0 events generate triggers on trigger output x.

XX1X: Channel 1 events generate triggers on trigger output x.

X1XX: Channel 2 events generate triggers on trigger output x.

1XXX: Channel 3 events generate triggers on trigger output x.

CTI trigger input status register (CTI_TRGISTSR)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: trigger input status

There is one bit of the register for each CTITRIGINx input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger output status register (CTI_TRGOSTSR)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: trigger output status

There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel input status register (CTI_CHINSTSR)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: channel input status

There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel output status register (CTI_CHOUTSTSR)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: channel output status

There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATER)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: channel output enable

For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.

0000: Channels events do not propagate.

XXX1: Channel 0 events propagate.

XX1X: Channel 1 events propagate.

X1XX: Channel 2 events propagate.

1XXX: Channel 3 events propagate.

CTI device configuration register (CTI_DEVIDR)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: number of ECT channels available
0x4: four channels

Bits 15:8 **NUMTRIG[7:0]**: number of ECT triggers available
0x8: height trigger inputs and height trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: number of trigger input/output multiplexers
0x0: none

CTI device type identifier register (CTI_DEVTYPER)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: sub-classification
0x1: cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight™ SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x21: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm® JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0xD: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

CTI CoreSight peripheral identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0x9: CoreSight™ component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

59.11.2 CTI register map

Table 738. CTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x004 to 0x00C	Reserved																																
0x010	CTI_INTACKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									X	X	X	X	X	X	X	X
0x014	CTI_APPSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR [3:0]			
	Reset value																													0	0	0	0

Table 738. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	CTI_APPPULSER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE [3:0]			
	Reset value																													X	X	X	X
0x020 to 0x03C	CTI_INEN0R to CTI_INEN7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN [3:0]			
	Reset value																													0	0	0	0
0x040 to 0x09C	Reserved																																
0x0A0 to 0x0BC	CTI_OUTEN0R to CTI_OUTEN7R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN [3:0]			
	Reset value																													0	0	0	0
0x0C0 to 0x12C	Reserved																																
0x130	CTI_TRGISTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRGOSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	CTI_CHINSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS [3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
	Reset value																													0	0	0	0
0x140	CTI_GATER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144 to 0xFC4	Reserved																																
0xFC8	CTI_DEVIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM [4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	CTI_DEVTYPER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	0	0	1
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1

Table 738. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Table 727: Processor ROM table* for register boundary addresses.

59.12 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers that control the power and clock behavior in debug mode. It allows the debugger (or the software) to:

- Maintain the clock and power to the process cores when in low-power modes (Sleep, Stop Stop, or Standby)
- Maintain the clock and power to the system debug and trace components when in low-power modes
- stop the clock to certain peripherals (SMBUS timeout, watchdogs, timers, RTC) when either processor core is stopped in the debug mode

59.12.1 Device ID

The DBGMCU includes an identity code register, DBGMCU_IDCODE. This register contains the ID code for the device. Debug tools can locate this register via the CoreSight™ discovery procedure described in *Section 59.5*.

59.12.2 Low-power mode emulation

When the device enters either Stop mode (clocks are stopped) or Standby mode (core power is switched off), the debugger can no longer access the debug access port and loses the connection with the device. To avoid this, the debugger (or software) can set the DBG_STANDBY and/or DBG_STOP bits in the *DBGMCU configuration register (DBGMCU_CR)*. These bits, when set, maintain the clock and power to the processor while the device is in the corresponding low-power mode. The processor remains in Sleep mode, and exits the low-power mode in the normal way. However, peripheral devices continue to operate, so the device behavior may not be identical to that of the actual low-power mode.

59.12.3 Peripheral clock freeze

The DBGMCU peripheral clock freeze registers allow to suspend operation of certain peripherals in debug mode. The peripherals supporting this feature are listed in *Table 739*.

Table 739. Peripheral clock freeze control bits

Bus	Control register	Peripheral	Description
APB1L	DBGMCU_APB1LFZR	I3C1	I3C1 SCL stall timeout counter
		I2C2	I2C2 SMBUS timeout
		I2C1	I2C1 SMBUS timeout
		IWDG	Independent watchdog
		WWDG	Window watchdog
		TIM14	General purpose timer 14
		TIM13	General purpose timer 13
		TIM12	General purpose timer 12
		TIM7	General purpose timer 7
		TIM6	General purpose timer 6
		TIM5	General purpose timer 5
		TIM4	General purpose timer 4
		TIM3	General purpose timer 3
		TIM2	General purpose timer 2
APB1H	DGBMCU_APB1HFZR	LPTIM2	Low power timer 2
APB2	DBGMCU_APB2FZR	TIM17	General purpose timer 17
		TIM16	General purpose timer 16
		TIM15	General purpose timer 15
		TIM8	General purpose timer 8
		TIM1	General purpose timer 1
APB3	DBGMCU_APB3FZR	RTC	Real time clock
		LPTIM6	Low power timer 6
		LPTIM5	Low power timer 5
		LPTIM4	Low power timer 4
		LPTIM3	Low power timer 3
		LPTIM1	Low power timer 1
		I2C4	I2C4 SMBUS timeout
		I2C3	I2C3 SMBUS timeout
AHB1	DBGMCU_AHB1FZR	GPDMA2 0 to 7	General purpose DMA2 channels 0 to 7
		GPDMA1 0 to 7	General purpose DMA1 channels 0 to 7

Each peripheral unit or DMA channel has a corresponding control bit, DBG_xxx_STOP, where xxx is the acronym of the peripheral (or DMA channel). The control bits are organized in DBGMCU_zzzFZR registers, where zzz corresponds to the name of the bus (AHB or APB). For example, DBGMCU_APB1LFZR contains the control bits for peripherals on the APB1L bus.

The control bit, when set, causes the corresponding peripheral operation to be suspended when the CPU is stopped in debug (HALTED = 1), according to the table below:

Table 740. Peripheral behaviour in debug mode

HALTED	DBG_xxx_STOP	Peripheral behaviour
0	X	The operation continues.
1	0	The operation continues.
1	1	The operation is suspended.

The accessibility of the bits DBG_xxx_STOP by the debugger depends on the state of the authentication signal spiden.

When spiden = 1 (secure privilege debug enabled), all bits can be modified and read by both debugger and software (secure or nonsecure).

When spiden = 0 (secure privilege debug disabled), only bits corresponding to nonsecure peripherals (or DMA channels) can be modified by the debugger or software. All bits can be read.

The status (secure or nonsecure) of a TrustZone–aware peripheral or a DMA channel, is signaled to the DBGMCU by the peripheral.

59.12.4 DBGMCU registers

The DBGMCU registers are reset only by a power-on reset.

They are accessible to the debugger via the AHB access port at base address 0xE00E 4000, and to software at base address 0x4402 4000.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x00

Reset value: 0xXXXX 6XXX

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Revision
A: 0x1000
Z: 0x1001
X: 0x1007

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identification
0x484: STM32H562/563/573
0x478: STM32H523/533

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x04

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, debugger accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCRT
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE [1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
								rw	rw	rw	rw		rw	rw	

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DCRT**: Debug credentials reset type
This bit selects which type of reset is used to revoke the debug authentication credentials
0: System reset
1: Power reset

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **TRACE_MODE[1:0]**: trace pin assignment
0x0: trace pins assigned for asynchronous mode (TRACESWO)
0x1: trace pins assigned for synchronous mode with a port width of 1 (TRACECK, TRACED0)
0x2: trace pins assigned for synchronous mode with a port width of 2 ((TRACECK, TRACED0-1)
0x3: trace pins assigned for synchronous mode with a port width of 4 ((TRACECK, TRACED0-3)

Bit 5 **TRACE_EN**: trace port and clock enable.
This bit enables the trace port clock, TRACECK.
0: disabled
1: enabled

Bit 4 **TRACE_IOEN**: trace pin enable
0: disabled - trace pins not assigned
1: enabled - trace pins assigned according to the value of TRACE_MODE field

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Allows debug in Standby mode
0: normal operation
All clocks are disabled and the core powered down automatically in Standby mode.
1: automatic clock stop/power down disabled
All active clocks and oscillators continue to run during Standby mode, and the core supply is maintained, allowing full debug capability. On exit from Standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Allows debug in Stop mode
0: normal operation
All clocks are disabled automatically in Stop mode.
1: automatic clock stop disabled
All active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0 Reserved, must be kept at reset value.

DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZR)

Address offset: 0x08

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
								rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
			rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_I3C1_STOP**: I3C1 SCL stall counter stop in debug
0: normal operation. I3C1 SCL stall timeout counter continues to operate while CPU is in debug mode.
1: stop in debug. I3C1 SCL stall timeout counter is frozen while CPU is in debug mode.

Bit 22 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug
0: normal operation. I2C2 SMBUS timeout continues to operate while CPU is in debug mode.
1: stop in debug. I2C2 SMBUS timeout is frozen while CPU is in debug mode.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug
0: normal operation. I2C1 SMBUS timeout continues to operate while CPU is in debug mode.
1: stop in debug. I2C1 SMBUS timeout is frozen while CPU is in debug mode.

Bits 20:13 Reserved, must be kept at reset value.

Bit 12 **DBG_IWDG_STOP**: IWDG stop in debug
0: normal operation. IWDG continues to operate while CPU is in debug mode.
1: stop in debug. IWDG is frozen while CPU is in debug mode.

Bit 11 **DBG_WWDG_STOP**: WWDG stop in debug
0: normal operation. WWDG continues to operate while CPU is in debug mode.
1: stop in debug. WWDG is frozen while CPU is in debug mode.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **DBG_TIM14_STOP**: TIM14 stop in debug
0: normal operation. TIM14 continues to operate while CPU is in debug mode.
1: stop in debug. TIM14 is frozen while CPU is in debug mode.

Bit 7 **DBG_TIM13_STOP**: TIM13 stop in debug
0: normal operation. TIM13 continues to operate while CPU is in debug mode.
1: stop in debug. TIM13 is frozen while CPU is in debug mode.

Bit 6 **DBG_TIM12_STOP**: TIM12 stop in debug
0: normal operation. TIM12 continues to operate while CPU is in debug mode.
1: stop in debug. TIM12 is frozen while CPU is in debug mode.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: normal operation. TIM7 continues to operate while CPU is in debug mode.
1: stop in debug. TIM7 is frozen while CPU is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: normal operation. TIM6 continues to operate while CPU is in debug mode.
1: stop in debug. TIM6 is frozen while CPU is in debug mode.

Bit 3 **DBG_TIM5_STOP**: TIM5 stop in debug
0: normal operation. TIM5 continues to operate while CPU is in debug mode.
1: Stop in debug. TIM5 is frozen while CPU is in debug mode.

Bit 2 **DBG_TIM4_STOP**: TIM4 stop in debug
0: normal operation. TIM4 continues to operate while CPU is in debug mode.
1: stop in debug. TIM4 is frozen while CPU is in debug mode.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: normal operation. TIM3 continues to operate while CPU is in debug mode.
1: stop in debug. TIM3 is frozen while CPU is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug
0: normal operation. TIM2 continues to operate while CPU is in debug mode.
1: stop in debug. TIM2 is frozen while CPU is in debug mode.

DBGMCU APB1H peripheral freeze register (DBGMCU_APB1HFZR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.
										rw					

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **DBG_LPTIM2_STOP**: LPTIM2 stop in debug
0: normal operation. LPTIM2 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM2 is frozen while CPU is in debug mode.

Bits 4:0 Reserved, must be kept at reset value.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x10

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP
													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw		rw											

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in debug
0: normal operation. TIM17 continues to operate while CPU is in debug mode.
1: stop in debug. TIM17 is frozen while CPU is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in debug
0: normal operation. TIM16 continues to operate while CPU is in debug mode.
1: stop in debug. TIM16 is frozen while CPU is in debug mode.

Bit 16 **DBG_TIM15_STOP**: TIM15 stop in debug
0: normal operation. TIM15 continues to operate while CPU is in debug mode.
1: stop in debug. TIM15 is frozen while CPU is in debug mode.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **DBG_TIM8_STOP**: TIM8 stop in debug
0: normal operation. TIM8 continues to operate while CPU is in debug mode.
1: stop in debug. TIM8 is frozen while CPU is in debug mode.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **DBG_TIM1_STOP**: TIM1 stop in debug
0: normal operation. TIM1 continues to operate while CPU is in debug mode.
1: stop in debug. TIM1 is frozen while CPU is in debug mode.

Bits 10:0 Reserved, must be kept at reset value.

DBGMCU APB3 peripheral freeze register (DBGMCU_APB3FZR)

Address offset: 0x14

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM6_STOP	DBG_LPTIM5_STOP	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM1_STOP	Res.
	rw									rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DBG_I2C4_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw										

Bit 31 Reserved, must be kept at reset value.

Bit 30 **DBG_RTC_STOP**: RTC stop in debug
0: normal operation. RTC continues to operate while CPU is in debug mode.
1: stop in debug. RTC is frozen while CPU is in debug mode.

Bits 29:22 Reserved, must be kept at reset value.

Bit 21 **DBG_LPTIM6_STOP**: LPTIM6 stop in debug
0: normal operation. LPTIM6 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM6 is frozen while CPU is in debug mode.

Bit 20 **DBG_LPTIM5_STOP**: LPTIM5 stop in debug
0: normal operation. LPTIM5 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM5 is frozen while CPU is in debug mode.

Bit 19 **DBG_LPTIM4_STOP**: LPTIM4 stop in debug
0: normal operation. LPTIM4 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM4 is frozen while CPU is in debug mode.

Bit 18 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug
0: normal operation. LPTIM3 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM3 is frozen while CPU is in debug mode.

Bit 17 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: normal operation. LPTIM1 continues to operate while CPU is in debug mode.
1: stop in debug. LPTIM1 is frozen while CPU is in debug mode.

Bits 16:12 Reserved, must be kept at reset value.

Bit 11 **DBG_I2C4_STOP**: I2C4 SMBUS timeout stop in debug
0: normal operation. I2C4 SMBUS timeout counter continues to operate while CPU is in debug mode.
1: stop in debug. I2C4 SMBUS timeout counter is frozen while CPU is in debug mode.

Bit 10 **DBG_I2C3_STOP**: I2C3 SMBUS timeout stop in debug
0: normal operation. I2C3 SMBUS timeout counter continues to operate while CPU is in debug mode.
1: stop in debug. I2C3 SMBUS timeout counter is frozen while CPU is in debug mode.

Bits 9:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x20

Reset value: 0x0000 0000

This register is accessible to the debugger and to the CPU after successful authentication. Prior to this, accesses are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA2_7_STOP	DBG_GPDMA2_6_STOP	DBG_GPDMA2_5_STOP	DBG_GPDMA2_4_STOP	DBG_GPDMA2_3_STOP	DBG_GPDMA2_2_STOP	DBG_GPDMA2_1_STOP	DBG_GPDMA2_0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA1_7_STOP	DBG_GPDMA1_6_STOP	DBG_GPDMA1_5_STOP	DBG_GPDMA1_4_STOP	DBG_GPDMA1_3_STOP	DBG_GPDMA1_2_STOP	DBG_GPDMA1_1_STOP	DBG_GPDMA1_0_STOP
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **DBG_GPDMA2_7_STOP**: GPDMA2 channel 7 stop in debug
0: normal operation. GPDMA2 channel 7 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 7 is frozen while CPU is in debug mode.

Bit 22 **DBG_GPDMA2_6_STOP**: GPDMA2 channel 6 stop in debug
0: normal operation. GPDMA2 channel 6 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 6 is frozen while CPU is in debug mode.

Bit 21 **DBG_GPDMA2_5_STOP**: GPDMA2 channel 5 stop in debug
0: normal operation. GPDMA2 channel 5 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 5 is frozen while CPU is in debug mode.

Bit 20 **DBG_GPDMA2_4_STOP**: GPDMA2 channel 4 stop in debug
0: normal operation. GPDMA2 channel 4 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 4 is frozen while CPU is in debug mode.

Bit 19 **DBG_GPDMA2_3_STOP**: GPDMA2 channel 3 stop in debug
0: normal operation. GPDMA2 channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 3 is frozen while CPU is in debug mode.

Bit 18 **DBG_GPDMA2_2_STOP**: GPDMA2 channel 2 stop in debug
0: normal operation. GPDMA2 channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 2 is frozen while CPU is in debug mode.

Bit 17 **DBG_GPDMA2_1_STOP**: GPDMA2 channel 1 stop in debug
0: normal operation. GPDMA2 channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 1 is frozen while CPU is in debug mode.

Bit 16 **DBG_GPDMA2_0_STOP**: GPDMA2 channel 0 stop in debug
0: normal operation. GPDMA2 channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA2 channel 0 is frozen while CPU is in debug mode.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DBG_GPDMA1_7_STOP**: GPDMA1 channel 7 stop in debug
0: normal operation. GPDMA1 channel 7 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 7 is frozen while CPU is in debug mode.

Bit 6 **DBG_GPDMA1_6_STOP**: GPDMA1 channel 6 stop in debug
0: normal operation. GPDMA1 channel 6 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 6 is frozen while CPU is in debug mode.

Bit 5 **DBG_GPDMA1_5_STOP**: GPDMA1 channel 5 stop in debug
0: normal operation. GPDMA1 channel 5 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 5 is frozen while CPU is in debug mode.

Bit 4 **DBG_GPDMA1_4_STOP**: GPDMA1 channel 4 stop in debug
0: normal operation. GPDMA1 channel 4 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 4 is frozen while CPU is in debug mode.

Bit 3 **DBG_GPDMA1_3_STOP**: GPDMA1 channel 3 stop in debug
0: normal operation. GPDMA1 channel 3 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 3 is frozen while CPU is in debug mode.

Bit 2 **DBG_GPDMA1_2_STOP**: GPDMA1 channel 2 stop in debug
0: normal operation. GPDMA1 channel 2 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 2 is frozen while CPU is in debug mode.

Bit 1 **DBG_GPDMA1_1_STOP**: GPDMA1 channel 1 stop in debug
0: normal operation. GPDMA1 channel 1 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 1 is frozen while CPU is in debug mode.

Bit 0 **DBG_GPDMA1_0_STOP**: GPDMA1 channel 0 stop in debug
0: normal operation. GPDMA1 channel 0 continues to operate while CPU is in debug mode.
1: stop in debug. GPDMA1 channel 0 is frozen while CPU is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0xFC

Reset value: 0x0001 XX03

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AP_ENABLED[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AP_PRESENT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
Bit n = 0: APn locked
Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in device
Bit n = 0: APn absent
Bit n = 1: APn present

DBGMCU debug authentication mailbox host register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0xXXXX XXXX

This register is read only when accessed by the CPU, writes have no effect.

This register can be written and read by an external debugger when system reset is asserted, or when access is granted during the authentication process.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Debug host to device mailbox message.
The debug host requests authentication by writing a value to this register prior to releasing the system reset. During debug authentication the debug host communicates with the device (CPU) via this register.

DBGMCU debug authentication mailbox device register (DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104

Reset value: 0xXXXX XXXX

This register is read only when accessed via the debug port, writes have no effect.

This register can be written and read by the CPU.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Device to debug host mailbox message.
During debug authentication the device (CPU) communicates with the debug host via this register.

DBGMCU debug authentication mailbox acknowledge register (DBGMCU_DBG_AUTH_ACK)

Address offset: 0x108

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEV_ACK	HOST_ACK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEV_ACK**: Device to host acknowledge.

This bit is set by hardware when the device (CPU) writes a message in the DBGMCU_DBG_AUTH_DEVICE register. It is reset automatically when the host (debugger) reads the message.

0: DBGMCU_DBG_AUTH_DEVICE register is empty

1. DBGMCU_DBG_AUTH_DEVICE register contains an unread message

Bit 0 **HOST_ACK**: Host to device acknowledge.

This bit is set by hardware when the host (debugger) writes a message in the DBGMCU_DBG_AUTH_HOST register. It is reset automatically when the device (CPU) reads the message.

0: DBGMCU_DBG_AUTH_HOST register is empty

1. DBGMCU_DBG_AUTH_HOST register contains an unread message

DBGMCU CoreSight peripheral identity register 4 (DBGMCU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: register file size
0x0: The register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 0 (DBGMCU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: part number bits [7:0]
0x00: DBGMCU part number

DBGMCU CoreSight peripheral identity register 1 (DBGMCU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0000

This register is always accessible

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x0: DBGMCU part number

DBGMCU CoreSight peripheral identity register 2 (DBGMCU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
0x1: designer identification specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x2: STMicroelectronics JEDEC code

DBGMCU CoreSight peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: metal fix version
0x0: no metal fix

Bits 3:0 **CMOD[3:0]**: customer modified
0x0: no customer modifications

DBGMCU CoreSight component identity register 0 (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component identification bits [7:0]
0x0D: common identification value

DBGMCU CoreSight component identity register 1 (DBGMCU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: component identification bits [15:12] - component class
0xF: Non-CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: component identification bits [11:8]
0x0: common identification value

DBGMCU CoreSight component identity register 2 (DBGMCU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component identification bits [23:16]
0x05: common identification value

DBGMCU CoreSight component identity register 3 (DBGMCU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

This register is always accessible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: component identification bits [31:24]
0xB1: common identification value

59.12.5 DBGMCU register map

Table 741. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x004	DBGMCU_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACE_MODE [1:0]		TRACE_EN	TRACE_IOEN	Res.	DBG_STANDBY	DBG_STOP	Res.
	Reset value																0									0	0	0	0		0	0	

Table 741. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	DBGMCU_ APB1LFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C1_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	DBG_WWDG_STOP	Res.	Res.	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value									0	0	0									0	0			0	0	0	0	0	0	0	0	0
0x0C0	DBGMCU_ APB1HFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM2_STOP	Res.	Res.	Res.	Res.	Res.
	Reset value																											0					
0x010	DBGMCU_ APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	Res.	Res.	DBG_TIM8_STOP	Res.	DBG_TIM1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0			0		0											
0x04	DBGMCU_ APB3FZR	Res.	DBG_RTC_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_LPTIM6_STOP	DBG_LPTIM5_STOP	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM1_STOP	Res.	Res.	Res.	Res.	Res.	DBG_I2C4_STOP	DBG_I2C3_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0									0	0	0	0	0						0	0											
0x018 to 0x01C	Reserved																																
0x020	DBGMCU_ AHB1FZR	Res.	Res.	Res.	Res.	Res	Res	Res	Res.	DBG_GPDMA2_7_STOP	DBG_GPDMA2_6_STOP	DBG_GPDMA2_5_STOP	DBG_GPDMA2_4_STOP	DBG_GPDMA2_3_STOP	DBG_GPDMA2_2_STOP	DBG_GPDMA2_1_STOP	DBG_GPDMA2_0_STOP	Res	Res	Res	Res	Res	Res	Res	Res	DBG_GPDMA1_7_STOP	DBG_GPDMA1_6_STOP	DBG_GPDMA1_5_STOP	DBG_GPDMA1_4_STOP	DBG_GPDMA1_3_STOP	DBG_GPDMA1_2_STOP	DBG_GPDMA1_1_STOP	DBG_GPDMA1_0_STOP
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x024 to 0x0F8	Reserved																																
0x0FC	DBGMCU_SR	AP_ENABLED[15:0]																AP_PRESENT[15:0]																
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x100	DBGMCU_DBG_ AUTH_HOST	MESSAGE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x104	DBGMCU_DBG_ AUTH_DEVICE	MESSAGE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 741. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x108	DBGMCU_DBG_AUTH_ACK	Res.																														DEV_ACK	HOST_ACK
	Reset value																															0	0
0x10C to 0xFBC	Reserved																																
0xFD0	DBGMCU_PIDR4	Res.																								SIZE [3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4 to 0xFDC	Reserved																																
0xFE0	DBGMCU_PIDR0	Res.																								PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xFE4	DBGMCU_PIDR1	Res.																								JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	0	0	0	0
0xFE8	DBGMCU_PIDR2	Res.																								REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	DBGMCU_PIDR3	Res.																								REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DBGMCU_CIDR0	Res.																								PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DBGMCU_CIDR1	Res.																								CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	1	0	0	0	0
0xFF8	DBGMCU_CIDR2	Res.																								PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DBGMCU_CIDR3	Res.																								PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

Refer to *Section 2.3* for register boundary addresses.

59.13 References

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G, 16th March 2015
3. DDI 0314H - Arm® CoreSight™ Components Technical Reference Manual, Issue H, 10 July, 2009
4. DDI 0553A (ID092917) - Arm® v8-M Architecture Reference Manual, Issue A.f, 29 September 2017
5. 100230_0002_00_en - Arm® Cortex®-M33 Processor r0p2 Technical Reference Manual, Issue 0002-00, 10 May 2017
6. 100232_0001_00_en - Arm® CoreSight™ ETM-M33 r0p1 Technical Reference Manual, Issue 0001-00, 3 February 2017

60 Device electronic signature

The device electronic signature is stored in the system memory area of the Flash memory module and can be read using the debug interface or by the CPU. It contains factory-programmed identification and calibration data that allow the user firmware or other external devices to automatically match to the characteristics of the devices.

60.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:

- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as part of the security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits cannot be altered by the user.

Base address: 0x08FF F800

Address offset: 0x00

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** X and Y coordinates on the wafer

Address offset: 0x04

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:8 **UID[63:40]:** LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 7:0 **UID[39:32]:** WAF_NUM[7:0]
Wafer number (8-bit unsigned number)

Address offset: 0x08

Read only = 0xXXXX XXXX where X is factory-programmed

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** LOT_NUM[55:24]
Lot number (ASCII encoded)

60.2 Flash size data register

Base address: 0x08FF F80C

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLASH_SIZE															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 15:0 **FLASH_SIZE[15:0]**: Flash memory size
This field indicates the size of the device Flash memory expressed in Kbytes.
As an example, 0x800 corresponds to 2048 Kbytes.

60.3 Package data register

Base address: 0x08FF F80E

Address offset: 0x00

Read only = 0xXXXX where X is factory-programmed

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[4:0]				
											r	r	r	r	r

Bits 15:5 Reserved, must be kept at reset value.

Bits 4:0 **PKG[4:0]**: Package type
00000: LQFP64
00001: VFQFPN68
00010: LQFP100
00011: UFBGA176+25
00100: LQFP144
00101: LQFP48
00110: UFBGA169
00111: LQFP176
01010: LQFP100 SMPS
01011: UFBGA176+25 SMPS
01100: LQFP144 SMPS
01101: LQFP176 SMPS
01110: UFBGA169 SMPS
10000: UFQFPN48
10001: WLCSP39
10100: UFBGA100
10101: UFBGA144
Others: reserved

Note: Refer to the product datasheet for availability of packages on a specific device.

61 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

62 Revision history

Table 742. Document revision history

Date	Revision	Changes
28-Feb-2023	1	Initial release.
29-Mar-2024	2	Document scope extended to STM32H523/533xx devices. Updated *Section 2.3.3: Embedded SRAMs*, *Section 5.6.8: GTZC1 TZSC memory x subregion A watermark configuration register (GTZC1_TZSC_MPCWMxACFGR)*, *DataProvisioning*, *Section 7.4.5: Description of data protection option bytes*, *Section 7.4.6: Description of boot address option bytes*, *Section 7.7.2: RSS user functions*, *Section 10.2: PWR main features*, *Clock security system on LSE*, *Section 8.4: ICACHE functional description*, *Section 8.4.7: Address remapping*, *Section 23.4.14: OCTOSPI device configuration*, *Section 48.7: I2C DMA requests*, *Section 32.7.1: RNG control register (RNG_CR)*, *Section 43.4.20: DMA requests*, *Section 43.7.3: LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3, 5, 6)*, *Section 46.3.11: RTC initialization and configuration*, *Section 46.6.20: RTC status register (RTC_SR)*, *Section 55.6.2: USBSRAM registers*, and *Debug authentication provisioning*. Added *Section 5.6.10: GTZC1 TZSC memory x subregion B watermark configuration register (GTZC1_TZSC_MPCWMxBCFGR)*, *Section 48.7.1: Transmission using DMA*, *Section 48.7.1: Transmission using DMA*, *Section 25.3: DLYB implementation*, *Section 36.3: PKA implementation*, *Memory types*, *Section 50.8.5: USART control register 3 [alternate] (USART_CR3)*, *Section 50.8.7: USART guard time and prescaler register (USART_GTPR)*, *Section 51.7.5: LPUART control register 3 [alternate] (LPUART_CR3)*, *Section 55.4.2: USB pins and internal signals*, and *Section 55.4.3: USB reset and clocks*. Added notes to *Section 46.6.21: RTC nonsecure masked interrupt status register (RTC_MISR)* and to *Section 46.6.22: RTC secure masked interrupt status register (RTC_SMISR)*. Updated *Figure 2: Memory map based on IDAU mapping*, *Figure 25: Flash high-cycle data memory map on 2-Mbyte devices*, *Figure 26: Flash high-cycle data memory map on 1-Mbyte devices*, *Figure 48: Clock tree*, *Figure 147: HyperBus read operation page crossing with latency*, *Figure 283: PSSI block diagram*, *Figure 284: Top-level block diagram*, *Figure 626: I2C initialization flow*, *Figure 629: Slave initialization flow*, *Figure 782: USB peripheral block diagram*, and *Figure 796: DMA transmission flow (standard mode)*. Added *Figure 27: Flash high-cycle data memory map on 512-Kbyte devices* and *Figure 27: Flash high-cycle data memory map on 512-Kbyte devices*. Updated *Table 3: Memory map and peripheral register addresses*, *Table 43: Specific OB modifying rules*, *Table 62: Product states, debug states and debug policy*, *Table 69: Macros for RSS services*, *Table 70: RSS lib interface functions*, *Table 75: Register map and reset value table*, *Table 79: RSSLIB/NSSLIB entry point access*, *Table 203: OCTOSPI implementation*, *Table 291: PSSI input/output pins*, *Table 298: RNG interrupt requests*, and *Table 299: RNG configurations*.Added *Table 78: RSSLIB/NSSLIB accesses*. Minor text edits across the whole document.

Table 742. Document revision history (continued)

Date	Revision	Changes
26-Mar-2025	3	Updated *Section 3.7.1: Temporal isolation using secure hide protection (HDP)*, *Section 3.11.2: Life cycle management*, *Flash memory architecture*, *Section 11.8.44: RCC kernel clock configuration register (RCC_CCIPR5)*, *Section 11.8.46: RCC reset status register (RCC_RSR)*, *Section 14.5.4: SBS debug control register (SBS_DBGCR)*, *Section 14.5.9: SBS product mode and configuration register (SBS_PMCR)*, *Section 15.3.5: Clock sources to timers*, *Section 18.6.18: EXTI external interrupt selection register (EXTI_EXTICR4)*, *SDRAM timing register x (FMC_SDTRx)*, *Section 59.5: ROM tables*, and *Section 59.12.4: DBGMCU registers*. Replaced SQRx and JSQRx by ADC_SQRy and ADC_JSQR registers, when referring to registers, JSQRi and SQRi registers by JSQi and SQi bits, when referring to bits in *Section 26: Analog-to-digital converters (ADC1/2)*. Updated *Table 49: Memory map and swapping options (STM32H523/533xx devices)*, *Table 51: Option bytes organization*, *Table 85: Register map and reset value table*, *Table 133: Peripherals interconnect matrix*, *Table 146: STM32H562/563/573xx vector table*, *Table 147: STM32H523/533xx vector table*, and *Table 548: Instance implementation*. Updated *Figure 52: Clock tree*, *Figure 132: Burst write SDRAM access waveforms*, *Figure 133: Burst read SDRAM access*, *Figure 135: Read access crossing row boundary*, and *Figure 136: Write access crossing row boundary*. Updated footnote *4* of *Table 115: Kernel clock distribution overview*, and added footnote *2* to *Table 150: EXTI line connections*.
23-Apr-2025	4	Updated bit description in *SDRAM timing register x (FMC_SDTRx)*.

Index

A

ADC_AWD2CR 1137
ADC_AWD3CR 1137
ADC_CALFACT 1139
ADC_CCR 1142
ADC_CDR 1145
ADC_CFGR 1117
ADC_CFGR2 1122
ADC_CR 1114
ADC_CSR 1140
ADC_DIFSEL 1138
ADC_DR 1132
ADC_IER 1112
ADC_ISR 1110
ADC_JDRy 1136
ADC_JSQR 1132
ADC_OFRy 1135
ADC_OR 1139
ADC_SMPR1 1124
ADC_SMPR2 1125
ADC_SQR1 1128
ADC_SQR2 1129
ADC_SQR3 1130
ADC_SQR4 1131
ADC_TR1 1126
ADC_TR2 1127
ADC_TR3 1128
AES_CR 1296
AES_DINR 1299
AES_DOUTR 1300
AES_ICR 1307
AES_IER 1305
AES_ISR 1306
AES_IVR0 1302
AES_IVR1 1302
AES_IVR2 1302
AES_IVR3 1303
AES_KEYR0 1300
AES_KEYR1 1301
AES_KEYR2 1301
AES_KEYR3 1301
AES_KEYR4 1303
AES_KEYR5 1303
AES_KEYR6 1304
AES_KEYR7 1304
AES_SR 1298
AES_SUSPxR 1304
AP0_CSWR 3004
AP1_CSWR 3005
APx_BASER 3007
APx_BDnR 3007
APx_DRWR 3007
APx_IDR 3008
APx_TAR 3006

B

BPU_CIDR0 3060
BPU_CIDR1 3060
BPU_CIDR2 3061
BPU_CIDR3 3061
BPU_COMPxR 3056
BPU_CTRLR 3056
BPU_DEVARCHR 3057
BPU_DEVTYPER 3057
BPU_PIDR0 3058
BPU_PIDR1 3059
BPU_PIDR2 3059
BPU_PIDR3 3060
BPU_PIDR4 3058

C

C1ROM_CIDR3 . . . . . 3017, 3022, 3028
CEC_CFGR 2975
CEC_CR 2974
CEC_IER 2979
CEC_ISR 2977
CEC_RXDR 2977
CEC_TXDR 2977
CORDIC_CSR 777
CORDIC_RDATA 780
CORDIC_WDATA 779
CPUROM_CIDR0 3026
CPUROM_CIDR1 3027
CPUROM_CIDR2 3027
CPUROM_CIDR3 3027
CPUROM_MEMTYPER 3023
CPUROM_PIDR0 3024
CPUROM_PIDR1 3025
CPUROM_PIDR2 3025
CPUROM_PIDR3 3026
CPUROM_PIDR4 3024
CRC_CR 760
CRC_DR 759
CRC_IDR 759
CRC_INIT 761

CRC_POL 761
CRS_CFGR 570
CRS_CR 568
CRS_ICR 573
CRS_ISR 571
CTI_APPCLEAR 3105
CTI_APPPULSER 3106
CTI_APPSETR 3105
CTI_CHINSTSR 3108
CTI_CHOUTSTSR 3109
CTI_CIDR0 3113
CTI_CIDR1 3113
CTI_CIDR2 3113
CTI_CIDR3 3114
CTI_CONTROLR 3104
CTI_DEVIDR 3110
CTI_DEVTYPER 3110
CTI_GATER 3109
CTI_INENxR 3106
CTI_INTACKR 3104
CTI_OUTENxR 3107
CTI_PIDR0 3111
CTI_PIDR1 3111
CTI_PIDR2 3112
CTI_PIDR3 3112
CTI_PIDR4 3110
CTI_TRGISTSR 3107
CTI_TRGOSTSR 3108

D

DAC_CCR 1199
DAC_CR 1188
DAC_DHR12L1 1193
DAC_DHR12L2 1194
DAC_DHR12LD 1196
DAC_DHR12R1 1192
DAC_DHR12R2 1194
DAC_DHR12RD 1195
DAC_DHR8R1 1193
DAC_DHR8R2 1195
DAC_DHR8RD 1196
DAC_DOR1 1197
DAC_DOR2 1197
DAC_MCR 1200
DAC_SHHR 1202
DAC_SHRR 1203
DAC_SHSR1 1201
DAC_SHSR2 1202
DAC_SR 1198
DAC_SWTRGR 1191
DBGMCU_AHB1FZR 3124
DBGMCU_APB1HFZR 3122
DBGMCU_APB1LFZR 3120
DBGMCU_APB2FZR 3122
DBGMCU_APB3FZR 3123
DBGMCU_CIDR0 3131
DBGMCU_CIDR1 3131
DBGMCU_CIDR2 3131
DBGMCU_CIDR3 3132
DBGMCU_CR 3119
DBGMCU_DBG_AUTH_ACK 3128
DBGMCU_DBG_AUTH_DEVICE 3127
DBGMCU_DBG_AUTH_HOST 3127
DBGMCU_IDCODE 3118
DBGMCU_PIDR0 3129
DBGMCU_PIDR1 3129
DBGMCU_PIDR2 3130
DBGMCU_PIDR3 3130
DBGMCU_PIDR4 3128
DBGMCU_SR 3126
DCACHE_CMDREADDRR 400
DCACHE_CMDRSADDRR 400
DCACHE_CR 395
DCACHE_FCR 398
DCACHE_IER 397
DCACHE_RHMONR 398
DCACHE_RMMONR 399
DCACHE_SR 396
DCACHE_WHMONR 399
DCACHE_WMMONR 399
DCMI_CR 1222
DCMI_CWSIZE 1231
DCMI_CWSTRT 1230
DCMI_DR 1231
DCMI_ESCR 1229
DCMI_ESUR 1229
DCMI_ICR 1228
DCMI_IER 1226
DCMI_MIS 1227
DCMI_RIS 1225
DCMI_SR 1225
DLYB_CFGR 1027
DLYB_CR 1026
DP_ABORTR 2996
DP_CTRLSTATR 2997
DP_DLCR 2998
DP_DLPIDR 2999
DP_DPIDR 2996
DP_RDBUFFR 3001
DP_RESENDR 3000
DP_SELECTR 3000
DP_TARGETIDR 2999
DTS_CFGR1 1158

DTS_DR 1161
DTS_ICIFR 1164
DTS_ITENR 1163
DTS_ITR1 1161
DTS_OR 1165
DTS_RAMPVALR 1160
DTS_SR 1162
DTS_T0VALR1 1160
DWT_CIDR0 3041
DWT_CIDR1 3042
DWT_CIDR2 3042
DWT_CIDR3 3043
DWT_COMPxR 3034
DWT_CPICNTR 3032
DWT_CTRLR 3030
DWT_CYCCNTR 3031
DWT_DEVARCHR 3038
DWT_DEVTYPER 3039
DWT_EXCCNTR 3032
DWT_FOLDCNTR 3033
DWT_FUNCTR0 3034
DWT_FUNCTR1 3035
DWT_FUNCTR2 3036
DWT_FUNCTR3 3037
DWT_LSUCNTR 3033
DWT_PCSR 3033
DWT_PIDR0 3040
DWT_PIDR1 3040
DWT_PIDR2 3040
DWT_PIDR3 3041
DWT_PIDR4 3039
DWT_SLPCNTR 3032

E

ETH_DMACCARXBR 2849
ETH_DMACCARXDR 2848
ETH_DMACCATXBR 2849
ETH_DMACCATXDR 2848
ETH_DMACCR 2836
ETH_DMACIER 2844
ETH_DMACMFCR 2852
ETH_DMACRXCR 2838
ETH_DMACRXDLAR 2841
ETH_DMACRXDTPR 2842
ETH_DMACRXIWTR 2847
ETH_DMACRXRLR 2843
ETH_DMACSR 2849
ETH_DMACTXCR 2837
ETH_DMACTXDLAR 2840
ETH_DMACTXDTPR 2842
ETH_DMACTXRLR 2843
ETH_DMADSR 2835
ETH_DMAISR 2835
ETH_DMAMR 2832
ETH_DMASBMR 2833
ETH_MAC1USTCR 2896
ETH_MACA0HR 2910
ETH_MACACR 2941
ETH_MACARPAR 2909
ETH_MACATSNR 2942
ETH_MACATSSR 2942
ETH_MACAxHR 2911
ETH_MACAxLR 2910
ETH_MACCR 2868
ETH_MACCSRSWCR 2909
ETH_MACDR 2897
ETH_MACECR 2873
ETH_MACHT0R 2877
ETH_MACHT1R 2878
ETH_MACHWF0R 2898
ETH_MACHWF1R 2900
ETH_MACHWF2R 2903
ETH_MACHWF3R 2905
ETH_MACIER 2889
ETH_MACISR 2887
ETH_MACIVIR 2883
ETH_MACL3A00R 2925
ETH_MACL3A01R 2930
ETH_MACL3A10R 2925
ETH_MACL3A11R 2930
ETH_MACL3A20R 2926
ETH_MACL3A21R 2931
ETH_MACL3A30R 2926
ETH_MACL3A31R 2931
ETH_MACL3L4C0R 2922
ETH_MACL3L4C1R 2927
ETH_MACL4A0R 2924
ETH_MACL4A1R 2929
ETH_MACLCSR 2893
ETH_MACLETR 2896
ETH_MACLMIR 2951
ETH_MACLTCR 2895
ETH_MACMDIOAR 2906
ETH_MACMDIODR 2908
ETH_MACPCSR 2891
ETH_MACPFR 2874
ETH_MACPOCR 2949
ETH_MACPPSCR 2944, 2946
ETH_MACPPSIR 2948
ETH_MACPPSTTNR 2948
ETH_MACPPSTTSR 2947
ETH_MACPPSWR 2949
ETH_MACQTXFCR 2884

ETH_MACRWKPFR 2893
ETH_MACRXFCR 2886
ETH_MACRXTXSR 2890
ETH_MACSPI0R 2950
ETH_MACSPI1R 2951
ETH_MACSPI2R 2951
ETH_MACSSIR 2934
ETH_MACSTNR 2936
ETH_MACSTNUR 2937
ETH_MACSTSR 2936
ETH_MACSTSUR 2937
ETH_MACTSAR 2938
ETH_MACTSCR 2932
ETH_MACTSEACR 2943
ETH_MACTSECNR 2944
ETH_MACTSIACR 2943
ETH_MACTSICNR 2944
ETH_MACTSSR 2939
ETH_MACTXTSSNR 2940
ETH_MACTXTSSSR 2941
ETH_MACVHTR 2881
ETH_MACVIR 2882
ETH_MACVR 2897
ETH_MACVTR 2879
ETH_MACWTR 2876
ETH_MMC_CONTROL 2912
ETH_MMC_RX_INTERRUPT 2913
ETH_MMC_RX_INTERRUPT_MASK 2916
ETH_MMC_TX_INTERRUPT 2914
ETH_MMC_TX_INTERRUPT_MASK 2917
ETH_MTLISR 2857
ETH_MTLOMR 2856
ETH_MTLQICSR 2860
ETH_MTLRXQDR 2865
ETH_MTLRXQMPOCR 2864
ETH_MTLRXQOMR 2862
ETH_MTLTXQDR 2859
ETH_MTLTXQOMR 2857
ETH_MTLTXQUR 2858
ETH_RX_ALIGNMENT_ERROR_PACKETS . . . 2919
ETH_RX_CRC_ERROR_PACKETS 2919
ETH_RX_LPI_TRAN_CNTR 2921
ETH_RX_LPI_USEC_CNTR 2921
ETH_RX_UNICAST_PACKETS_GOOD . . . 2920
ETH_TX_LPI_TRAN_CNTR 2921
ETH_TX_LPI_USEC_CNTR 2920
ETH_TX_MULTIPLE_COLLISION_-GOOD_PACKETS 2918
ETH_TX_PACKET_COUNT_GOOD 2919
ETH_TX_SINGLE_COLLISION_GOOD_PACK-ETS 2918
ETM_AUTHSTATR 3082
ETM_CCCTLR 3067
ETM_CIDR0 3085
ETM_CIDR1 3086
ETM_CIDR2 3086
ETM_CIDR3 3087
ETM_CLAIMCLR 3081
ETM_CLAIMSETR 3081
ETM_CNTRLDVR0 3069
ETM_CONFIGR 3064
ETM_DEVARCHR 3082
ETM_DEVTYPER 3083
ETM_EVENTCTL0R 3065
ETM_EVENTCTL1R 3066
ETM_IDR0 3072
ETM_IDR1 3073
ETM_IDR10 3070
ETM_IDR11 3071
ETM_IDR12 3071
ETM_IDR13 3071
ETM_IDR2 3074
ETM_IDR3 3074
ETM_IDR4 3075
ETM_IDR5 3076
ETM_IDR8 3069
ETM_IDR9 3070
ETM_IMSPECR0 3072
ETM_PDCR 3079
ETM_PDSR 3080
ETM_PIDR0 3084
ETM_PIDR1 3084
ETM_PIDR2 3084
ETM_PIDR3 3085
ETM_PIDR4 3083
ETM_PRGCTLR 3063
ETM_RSCTLR2 3077
ETM_RSCTLR3 3077
ETM_SSCCR0 3078
ETM_SSCSR0 3078
ETM_SSPCICR0 3079
ETM_STALLCTLR 3066
ETM_STATR 3064
ETM_SYNCPR 3067
ETM_TRACEIDR 3068
ETM_VICTLR 3068
EXTI_EMR1 751
EXTI_EMR2 753
EXTI_EXTICR1 742
EXTI_EXTICR2 744
EXTI_EXTICR3 746
EXTI_EXTICR4 748
EXTI_FPR1 733

EXTI_FPR2 740
EXTI_FTSR1 731
EXTI_FTSR2 736
EXTI_IMR1 751
EXTI_IMR2 752
EXTI_LOCKR 750
EXTI_PRIVCFGR1 735
EXTI_PRIVCFGR2 742
EXTI_RPR1 733
EXTI_RPR2 739
EXTI_RTSR1 731
EXTI_RTSR2 735
EXTI_SECCFGR1 734
EXTI_SECCFGR2 741
EXTI_SWIER1 732
EXTI_SWIER2 737

F

FDCAN_CCCR 2571
FDCAN_CKDIV 2596
FDCAN_CREL 2568
FDCAN_DBTP 2568
FDCAN_ECR 2576
FDCAN_ENDN 2568
FDCAN_HPMS 2587
FDCAN_IE 2582
FDCAN_ILE 2585
FDCAN_ILS 2584
FDCAN_IR 2579
FDCAN_NBTP 2572
FDCAN_PSR 2577
FDCAN_RWD 2570
FDCAN_RXF0A 2589
FDCAN_RXF0S 2588
FDCAN_RXF1A 2590
FDCAN_RXF1S 2589
FDCAN_RXGFC 2585
FDCAN_TDCR 2579
FDCAN_TEST 2569
FDCAN_TOCC 2575
FDCAN_TOCV 2576
FDCAN_TSCC 2574
FDCAN_TSCV 2574
FDCAN_TXBAR 2592
FDCAN_TXBC 2590
FDCAN_TXBCF 2594
FDCAN_TXBCIE 2595
FDCAN_TXBCR 2593
FDCAN_TXBRP 2591
FDCAN_TXBTIE 2594
FDCAN_TXBTO 2593
FDCAN_TXEFA 2596
FDCAN_TXEFS 2595
FDCAN_TXFQS 2591
FDCAN_XIDAM 2587
FLASH_ACR 318
FLASH_BOOTR_PRG 345
FLASH_ECCCORR 352
FLASH_ECCDETR 353
FLASH_ECCDR 354
FLASH_EDATA1R_CUR 350
FLASH_EDATA1R_PRG 351
FLASH_EDATA2R_CUR 358
FLASH_EDATA2R_PRG 359
FLASH_HDP1R_CUR 351
FLASH_HDP1R_PRG 352
FLASH_HDP2R_CUR 359
FLASH_HDP2R_PRG 360
FLASH_HDPEXTR 336
FLASH_NSBOOTR_CUR 343
FLASH_NSBOOTR_PRG 344
FLASH_NSCCR 332
FLASH_NSCR 327
FLASH_NSEPOCHR_CUR 340
FLASH_NSKEYR 319
FLASH_NSOBKCFGR 334
FLASH_NSOBKKEYR 320
FLASH_NSSR 323
FLASH_OPSR 321
FLASH_OPTCR 322
FLASH_OPTKEYR 320
FLASH_OPTSR_CUR 337
FLASH_OPTSR_PRG 339
FLASH_OPTSR2_CUR 341
FLASH_OPTSR2_PRG 342
FLASH_OTPBLR_CUR 346
FLASH_OTPBLR_PRG 346
FLASH_PRIVBB1Rx 347
FLASH_PRIVBB2Rx 355
FLASH_PRIVCFGR 334
FLASH_SECBB1Rx 347
FLASH_SECBB2Rx 355
FLASH_SECBOOTR_CUR 344
FLASH_SECCCR 333
FLASH_SECCR 329
FLASH_SECEPOCHR_CUR 341
FLASH_SECKEYR 319
FLASH_SECOBKCFGR 335
FLASH_SECOBKKEYR 320
FLASH_SECSR 325
FLASH_SECWM1R_CUR 348
FLASH_SECWM1R_PRG 348
FLASH_SECWM2R_CUR 356

FLASH_SECWM2R_PRG 356
FLASH_WRP1R_CUR 349
FLASH_WRP1R_PRG 349
FLASH_WRP2R_CUR 357
FLASH_WRP2R_PRG 358
FMAC_CR 804
FMAC_PARAM 803
FMAC_RDATA 807
FMAC_SR 805
FMAC_WDATA 806
FMAC_X1BUFCFG 801
FMAC_X2BUFCFG 801
FMAC_YBUFCFG 802
FMC_BCRx 847
FMC_BTRx 850
FMC_BWTRx 852
FMC_ECCR 865
FMC_PATT 864
FMC_PCR 860
FMC_PCSCNTR 854
FMC_PMEM 863
FMC_SDCMR 879
FMC_SDCRx 876
FMC_SDRTR 880
FMC_SDSR 882
FMC_SDTRx 877
FMC_SR 862

G

GPDMA_CxBR1 696-697
GPDMA_CxBR2 704
GPDMA_CxCR 686
GPDMA_CxDAR 702
GPDMA_CxFCR 684
GPDMA_CxLBAR 683
GPDMA_CxLLR 705, 707
GPDMA_CxSAR 700
GPDMA_CxSR 685
GPDMA_CxTR1 688
GPDMA_CxTR2 692
GPDMA_CxTR3 703
GPDMA_MISR 682
GPDMA_PRIVCFGR 681
GPDMA_RCFGLOCKR 681
GPDMA_SECCFGR 680
GPDMA_SMISR 683
GPIOx_AFRH 591
GPIOx_AFRL 590
GPIOx_BRR 592
GPIOx_BSRR 589
GPIOx_HSLVR 592
GPIOx_IDR 588
GPIOx_LCKR 589
GPIOx_MODER 586
GPIOx_ODR 588
GPIOx_OSPEEDR 587
GPIOx_OTYPER 586
GPIOx_PUPDR 587
GPIOx_SECCFGR 593
GTZC1_MPCBBz_CFGLOCK1 229
GTZC1_MPCBBz_CR 229
GTZC1_MPCBBz_PRIVCFGRx 230
GTZC1_MPCBBz_SECCFGRx 230
GTZC1_TZIC_FCR1 218
GTZC1_TZIC_FCR2 220
GTZC1_TZIC_FCR3 222
GTZC1_TZIC_FCR4 224
GTZC1_TZIC_IER1 201
GTZC1_TZIC_IER2 203
GTZC1_TZIC_IER3 205
GTZC1_TZIC_IER4 207
GTZC1_TZIC_SR1 209
GTZC1_TZIC_SR2 212
GTZC1_TZIC_SR3 214
GTZC1_TZIC_SR4 216
GTZC1_TZSC_CR 181
GTZC1_TZSC_MPCWMxACFGR 195-196
GTZC1_TZSC_MPCWMxAR 196
GTZC1_TZSC_MPCWMxBR 197
GTZC1_TZSC_PRIVCFGR1 188
GTZC1_TZSC_PRIVCFGR2 191
GTZC1_TZSC_PRIVCFGR3 193
GTZC1_TZSC_SECCFGR1 181
GTZC1_TZSC_SECCFGR2 184
GTZC1_TZSC_SECCFGR3 186

H

HASH_CR 1377
HASH_CSRx 1385
HASH_DIN 1379
HASH_HRAx 1382
HASH_HRx 1382
HASH_IMR 1383
HASH_SR 1383
HASH_STR 1380

I

I2C_CR1 2120
I2C_CR2 2123
I2C_ICR 2131
I2C_ISR 2128
I2C_OAR1 2125

I2C_OAR2 2125
I2C_PECR 2132
I2C_RXDR 2132
I2C_TIMEOUTR 2127
I2C_TIMINGR 2126
I2C_TXDR 2133
I3C_BCR 2234
I3C_CEVR 2222
I3C_CFGR 2201
I3C_CR 2197, 2199
I3C_CRCAPR 2237
I3C_DCR 2235
I3C_DEVR0 2224
I3C_DEVRx 2226
I3C_EPIDR 2240
I3C_EVR 2216
I3C_GETCAPR 2236
I3C_GETMXDSR 2238
I3C_IBIDR 2210
I3C_IER 2220
I3C_MAXRLR 2228
I3C_MAXWLR 2229
I3C_RDR 2206
I3C_RDWR 2206
I3C_RMR 2215
I3C_SER 2213
I3C_SR 2212
I3C_TDR 2207
I3C_TDWR 2208
I3C_TGTTDR 2211
I3C_TIMINGR0 2230
I3C_TIMINGR1 2231
I3C_TIMINGR2 2233
ICACHE_CR 377
ICACHE_CRRx 380
ICACHE_FCR 379
ICACHE_HMONR 380
ICACHE_IER 379
ICACHE_MMONR 380
ICACHE_SR 378
ITM_CIDR0 3052
ITM_CIDR1 3053
ITM_CIDR2 3053
ITM_CIDR3 3054
ITM_DEVARCHR 3049
ITM_DEVTYPER 3050
ITM_PIDR0 3051
ITM_PIDR1 3051
ITM_PIDR2 3051
ITM_PIDR3 3052
ITM_PIDR4 3050
ITM_STIMRx 3046
ITM_TCR 3048
ITM_TER 3047
ITM_TPR 3047
IWDG_EWCR 1954
IWDG_KR 1951
IWDG_PR 1951
IWDG_RLR 1952
IWDG_SR 1952
IWDG_WINR 1954

L

LPTIM_ARR 1933
LPTIM_CCMR1 1935
LPTIM_CCR1 1932
LPTIM_CCR2 1938
LPTIM_CFGR 1928
LPTIM_CFGR2 1934
LPTIM_CNT 1933
LPTIM_CR 1931
LPTIM_RCR 1935
LPTIM4_DIER 1923
LPTIM4_ICR 1920
LPTIM4_ISR 1915
LPTIMx_DIER 1925-1926
LPTIMx_ICR 1921-1922
LPTIMx_ISR 1916, 1918
LPUART_BRR 2377
LPUART_CR1 . . . . . 2364, 2367
LPUART_CR2 2370
LPUART_CR3 . . . . . 2372, 2375
LPUART_ICR 2386
LPUART_ISR 2378, 2383
LPUART_PRESC 2388
LPUART_RDR 2387
LPUART_RQR 2377
LPUART_TDR 2387

M

MCUROM_CIDR0 3021
MCUROM_CIDR1 3021
MCUROM_CIDR2 3022
MCUROM_CIDR3 3022
MCUROM_MEMTYPER 3018
MCUROM_PIDR0 3019
MCUROM_PIDR1 3019
MCUROM_PIDR2 3020
MCUROM_PIDR3 3020
MCUROM_PIDR4 3018

O

OCTOSPI_ABR 929
OCTOSPI_AR 924
OCTOSPI_CCR 926
OCTOSPI_CR 915
OCTOSPI_DCR1 918
OCTOSPI_DCR2 920
OCTOSPI_DCR3 921
OCTOSPI_DCR4 921
OCTOSPI_DLR 923
OCTOSPI_DR 924
OCTOSPI_FCR 923
OCTOSPI_HLCR 937
OCTOSPI_IR 929
OCTOSPI_LPTR 930
OCTOSPI_PIR 926
OCTOSPI_PSMAR 926
OCTOSPI_PSMKR 925
OCTOSPI_SR 922
OCTOSPI_TCR 928
OCTOSPI_WABR 937
OCTOSPI_WCCR 934
OCTOSPI_WIR 936
OCTOSPI_WPABR 933
OCTOSPI_WPCCR 930
OCTOSPI_WPIR 933
OCTOSPI_WPTCR 932
OCTOSPI_WTCR 936
OTFDEC_CR 1429
OTFDEC_ICR 1437
OTFDEC_IER 1438
OTFDEC_ISR 1436
OTFDEC_PRIVCFGR 1430
OTFDEC_RxCFGR 1430
OTFDEC_RxENDADDR 1432
OTFDEC_RxKEYR0 1434
OTFDEC_RxKEYR1 1435
OTFDEC_RxKEYR2 1435
OTFDEC_RxKEYR3 1436
OTFDEC_RxNONCER0 1433
OTFDEC_RxNONCER1 1434
OTFDEC_RxSTARTADDR 1432

P

PKA_CLRFR 1418
PKA_CR 1415
PKA_SR 1417
PSSI_CR 1241
PSSI_DR 1245
PSSI_ICR 1245
PSSI_IER 1244
PSSI_MIS 1244
PSSI_RIS 1243
PSSI_SR 1242
PWR_BDCR 439
PWR_BDSR 441
PWR_DBPCR 441
PWR_IORETR 447
PWR_PMCR 434, 436
PWR_PMSR 437
PWR_PRIVCFGR 449
PWR_SCCR 442
PWR_SECCFGR 448
PWR_UCPDR 442
PWR_USBSCR 444
PWR_VMCR 443
PWR_VMSR 445
PWR_VOSCR 438
PWR_VOSSR 439
PWR_WUCR 447
PWR_WUSCR 446
PWR_WUSR 446

R

RAMCFG_M2WPR1 240
RAMCFG_M2WPR2 240
RAMCFG_M2WPR3 241
RAMCFG_MxCR 237
RAMCFG_MxDEAR 239
RAMCFG_MxECCKEYR 241
RAMCFG_MxERKEYR 242
RAMCFG_MxICR 239
RAMCFG_MxIER 237
RAMCFG_MxISR 238
RAMCFG_MxSEAR 239
RCC_AHB1ENR 516
RCC_AHB1LPENR 528
RCC_AHB1RSTR 505
RCC_AHB2ENR 518
RCC_AHB2LPENR 530
RCC_AHB2RSTR 506
RCC_AHB4ENR 520
RCC_AHB4LPENR 532
RCC_AHB4RSTR 508
RCC_APB1HENR 524
RCC_APB1HLPENR 536
RCC_APB1HRSTR 512
RCC_APB1LENR 521
RCC_APB1LLPENR 533
RCC_APB1LRSTR 509
RCC_APB2ENR 525
RCC_APB2LPENR 537

RCC_APB2RSTR 513
RCC_APB3ENR 526
RCC_APB3LPENR 539
RCC_APB3RSTR 515
RCC_BDCR 551
RCC_CCIPR1 541
RCC_CCIPR2 543
RCC_CCIPR3 546
RCC_CCIPR4 547
RCC_CCIPR5 549
RCC_CFGR1 485
RCC_CFGR2 488
RCC_CICR 504
RCC_CIER 501
RCC_CIFR 502
RCC_CR 481
RCC_CRRCR 484
RCC_CSICFGR 485
RCC_HSICFGR 484
RCC_PLL1CFGR 490
RCC_PLL1DIVR 495
RCC_PLL1FRACR 497
RCC_PLL2CFGR 492
RCC_PLL2DIVR 497
RCC_PLL2FRACR 498
RCC_PLL3CFGR 494
RCC_PLL3DIVR 499
RCC_PLL3FRACR 500
RCC_PRIVCFGR 556
RCC_RSR 553
RCC_SECCFGR 554
RNG_CR 1258
RNG_DR 1261
RNG_HTCR 1263
RNG_NSCR 1262
RNG_SR 1260
RTC_ALRABINR 2017
RTC_ALRBBINR 2017
RTC_ALRMAR 2007
RTC_ALRMASSR 2009
RTC_ALRMBR 2010
RTC_ALRMBSSR 2011
RTC_CALR 2003
RTC_CR 1995
RTC_DR 1990
RTC_ICSR 1992
RTC_MISR 2013
RTC_OR 2016
RTC_PRER 1994
RTC_PRIVCFGR 1999
RTC_SCR 2015
RTC_SECCFGR 2001
RTC_SHIFTR 2004
RTC_SMISR 2014
RTC_SR 2012
RTC_SSR 1991
RTC_TR 1989
RTC_TSDR 2006
RTC_TSSSR 2007
RTC_TSTR 2005
RTC_WPR 2002
RTC_WUTR 1995

S

SAES_CR 1350
SAES_DINR 1354
SAES_DOUTR 1355
SAES_ICR 1362
SAES_IER 1360
SAES_ISR 1361
SAES_IVR0 1357
SAES_IVR1 1357
SAES_IVR2 1357
SAES_IVR3 1358
SAES_KEYR0 1355
SAES_KEYR1 1356
SAES_KEYR2 1356
SAES_KEYR3 1356
SAES_KEYR4 1358
SAES_KEYR5 1358
SAES_KEYR6 1359
SAES_KEYR7 1359
SAES_SR 1353
SAES_SUSPxR 1359
SAI_ACLRFR 2517
SAI_ACR1 2506
SAI_ACR2 2509
SAI_ADR 2518
SAI_AFRCR 2511
SAI_AIM 2513
SAI_ASLOTR 2512
SAI_ASR 2515
SAI_BCLRFR 2528
SAI_BCR1 2518
SAI_BCR2 2521
SAI_BDR 2529
SAI_BFRCR 2523
SAI_BIM 2525
SAI_BSLOTR 2524
SAI_BSR 2526
SAI_GCR 2506
SAI_PDMCR 2530
SAI_PDMDLY 2531

SBS_CCCSR 614
SBS_CCSWCR 616
SBS_CCVALR 615
SBS_CFGR2 616
SBS_CNSLCKR 617
SBS_CSLCKR 618
SBS_DBGCR 608
SBS_DBGLOCKR 609
SBS_ECCNMIR 619
SBS_EPOCHSELCR 610
SBS_FPUIMR 613
SBS_HDPLCR 607
SBS_HDPLSR 607
SBS_MESR 613
SBS_NEXTHDPLCR 608
SBS_PMCR 611
SBS_RSSCMDR 610
SBS_SECCFGR 611
SDMMC_ACKTIMER 1016
SDMMC_ARGR 1001
SDMMC_CLKCR 999
SDMMC_CMDR 1001
SDMMC_DCNTR 1007
SDMMC_DCTRL 1006
SDMMC_DLENR 1005
SDMMC_DTIMER 1004
SDMMC_FIFORx 1020
SDMMC_ICR 1011
SDMMC_IDMABAR 1019
SDMMC_IDMABASER 1018
SDMMC_IDMABSIZER 1017
SDMMC_IDMACTRLR 1016
SDMMC_IDMALAR 1018
SDMMC_MASKR 1013
SDMMC_POWER 998
SDMMC_RESPCMDR 1003
SDMMC_RESPxR 1004
SDMMC_STAR 1008
SPI_CFG1 2450
SPI_CFG2 2453
SPI_CR1 2447
SPI_CR2 2449
SPI_CRCPOLY 2461
SPI_I2SCFGR 2463
SPI_IER 2455
SPI_IFCR 2459
SPI_RXCRC 2462
SPI_RXDR 2460
SPI_SR 2456
SPI_TXCRC 2461
SPI_TXDR 2460
SPI_UDRDR 2463
SYSROM_CIDR0 3015
SYSROM_CIDR1 3016
SYSROM_CIDR2 3016
SYSROM_CIDR3 3016
SYSROM_MEMTYPER 3013
SYSROM_PIDR0 3013
SYSROM_PIDR1 3014
SYSROM_PIDR2 3014
SYSROM_PIDR3 3015
SYSROM_PIDR4 3013

T

TAMP_ATCR1 2047
TAMP_ATCR2 2051
TAMP_ATOR 2050
TAMP_ATSEEDR 2049
TAMP_BKPxR 2068
TAMP_COUNT1R 2066
TAMP_CR1 2039
TAMP_CR2 2041
TAMP_CR3 2044
TAMP_FLTCR 2045
TAMP_IER 2057
TAMP_MISR 2061
TAMP_OR 2066
TAMP_PRIVCFGR 2056
TAMP_RPCFGR 2067
TAMP_SCR 2064
TAMP_SECCFGR 2054
TAMP_SMISR 2062
TAMP_SR 2059
TIM12_ARR 1767
TIM12_CCER 1765
TIM12_CCMR1 1761-1762
TIM12_CCR1 1768
TIM12_CCR2 1768
TIM12_CNT 1766
TIM12_CR1 1753
TIM12_CR2 1755
TIM12_DIER 1758
TIM12_EGR 1760
TIM12_PSC 1767
TIM12_SMCR 1755
TIM12_SR 1759
TIM12_TISEL 1769
TIM15_AF1 1856
TIM15_AF2 1858
TIM15_ARR 1849
TIM15_BDTR 1851
TIM15_CCER 1845
TIM15_CCMR1 1841-1842

TIM15_CCR1 1850
TIM15_CCR2 1851
TIM15_CNT 1848
TIM15_CR1 1832
TIM15_CR2 1833
TIM15_DCR 1859
TIM15_DIER 1837
TIM15_DMAR 1860
TIM15_DTR2 1854
TIM15_EGR 1840
TIM15_PSC 1848
TIM15_RCR 1849
TIM15_SMCR 1835
TIM15_SR 1838
TIM15_TISEL 1855
TIM17_OR1 1884
TIMx_AF1 . . . . . 1575, 1695, 1881
TIMx_AF2 . . . . . 1578, 1696, 1884
TIMx_ARR 1561, 1685, 1719, 1780, 1875
TIMx_BDTR 1565, 1877
TIMx_CCER 1556, 1682, 1778, 1871
TIMx_CCMR1 . . 1547, 1549, 1674, 1676, 1775-1776, ... 1868-1869
TIMx_CCMR2 . . . . . 1552-1553, 1678-1679
TIMx_CCMR3 1571
TIMx_CCR1 . . . . . 1562, 1686-1687, 1781, 1876
TIMx_CCR2 . . . . . 1562, 1687-1688
TIMx_CCR3 . . . . . 1563, 1689-1690
TIMx_CCR4 . . . . . 1564, 1691-1692
TIMx_CCR5 1569
TIMx_CCR6 1570
TIMx_CNT . 1560, 1683-1684, 1718, 1779, 1874
TIMx_CR1 . . . . . 1533, 1663, 1715, 1772, 1863
TIMx_CR2 . . . . . 1534, 1664, 1717, 1864
TIMx_DCR 1580, 1697, 1885
TIMx_DIER 1542, 1670, 1717, 1773, 1865
TIMx_DMAR 1582, 1698, 1886
TIMx_DTR2 . . . . . 1572, 1880
TIMx_ECR 1573, 1693
TIMx_EGR 1546, 1673, 1718, 1774, 1867
TIMx_PSC 1560, 1684, 1719, 1780, 1874
TIMx_RCR 1561, 1875
TIMx_SMCR 1538, 1666
TIMx_SR 1543, 1671, 1718, 1773, 1866
TIMx_TISEL 1574, 1694, 1781, 1881
TPIU_ACPR 3092
TPIU_CIDR0 3099
TPIU_CIDR1 3100
TPIU_CIDR2 3100
TPIU_CIDR3 3101
TPIU_CLAIMCLR 3096
TPIU_CLAIMSETR 3095
TPIU_CSPSR 3092
TPIU_DEVIDR 3096
TPIU_DEVTYPER 3097
TPIU_FFCR 3094
TPIU_FFSR 3093
TPIU_PIDR0 3098
TPIU_PIDR1 3098
TPIU_PIDR2 3098
TPIU_PIDR3 3099
TPIU_PIDR4 3097
TPIU_PSCR 3095
TPIU_SPPR 3093
TPIU_SSPSR 3092

U

UCPD_CFGR1 2669
UCPD_CFGR2 2671
UCPD_CFGR3 2671
UCPD_CR 2672
UCPD_ICR 2679
UCPD_IMR 2674
UCPD_RX_ORDEXTR1 2683
UCPD_RX_ORDEXTR2 2684
UCPD_RX_ORDSETR 2681
UCPD_RX_PAYSZR 2682
UCPD_RXDR 2683
UCPD_SR 2676
UCPD_TX_ORDSETR 2680
UCPD_TX_PAYSZR 2680
UCPD_TXDR 2681
USART_BRR 2315
USART_CR1 . . . . . 2296, 2300
USART_CR2 2303
USART_CR3 . . . . . 2307, 2311
USART_GTPR 2315
USART_ICR 2329
USART_ISR 2318, 2324
USART_PRESC 2331
USART_RDR 2330
USART_RQR 2317
USART_RTOR 2316
USART_TDR 2331
USB_BCDR 2631
USB_CHEP_RXTXBD_n 2645-2646
USB_CHEP_TXRXBD_n 2643
USB_CHEPnR 2632
USB_CNTR 2622
USB_DADDR 2629
USB_FNR 2629
USB_ISTR 2625
USB_LPMCSR 2630

V

VREFBUF_CCR 1210
VREFBUF_CSR 1209

W

WWDG_CFR 1962
WWDG_CR 1961
WWDG_SR 1963

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved